Sun Java™ System

Messaging Server 6 Administration Reference

2004Q2

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

Part No: 817-6267-10

Contents

List of Tables

About This Guide

This manual provides reference information about the Beta version of Sun™ Java System Messaging Server product. Messaging Server provides a powerful and flexible cross-platform solution to the email needs of enterprises and messaging hosts of all sizes using open Internet standards.

Use this manual as a companion to the *Messaging Server Administration Guide*. The administrator's guide describes how to configure, maintain, monitor, and troubleshoot Messaging Server. This Administration Reference provides information about command-line utilities and configuration files. This information enables you to configure, maintain, monitor, and troubleshoot Messaging Server.

Topics covered in this chapter include:

Who Should Read This Book

This manual is intended for highly or moderately technical network administrators with experience in UNIX® or Windows NT. These administrators will be configuring, administering, and maintaining Messaging Server. Architects and developers may also use the *Messaging Server Administration Reference*. This manual is not intended for end users.

What You Need to Know

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

- The Internet and the World Wide Web
- Sun Java System Administration Server
- Sun Java System Directory Server and LDAP

How This Book is Organized

This book contains the following chapters:

- About This Guide (this chapter)
- Chapter 1, “Messaging Server Command-line Utilities”

  This chapter describes the core Messaging Server utilities.
- Chapter 2, “Message Transfer Agent Command-line Utilities”

  This chapter describes the MTA utilities.
- Chapter 3, “Messaging Server Configuration”

  This chapter lists the configuration parameters for the Messaging Server.
- Chapter 4, “MTA Configuration”

  This chapter describes the channel keywords, rewrite rule configuration, and MTA configuration files.
- Chapter 5, “Messaging Multiplexor Configuration”

This chapter describes the configuration files and configuration parameters for the Messaging Multiplexor.

- Appendix A, "Supported Standards"

  This appendix lists national, international, and industry standards related to electronic messaging and for which support is claimed by Messaging Server.

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.

Italicized Font

Italicized font is used to represent text that you enter using information that is unique to your messaging server. It is used for server paths and names and account IDs.

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

msg_svr_base`/...`

In these situations, *msg_svr_base* represents the directory path in which you install the server. For example, if you install your server in the directory `/opt/SUNWmsgsr`, then *msg_svr_base* refers to `/opt/SUNWmsgsr`.

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

```
imadmin admin remove -D login -l userid -n domain -w password [-d domain]
  [-h] [-i inputfile] [-p idaport] [-X idahost] [-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 Top-Level Administrator's password for *password* when you enter the `imadmin admin remove` command.

Square or Straight Brackets

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

`readership [-d` *days*`] [-p` *months*`]`

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

```
readership
```

However, the presence of `[-d` *days*`]` and `[-p` *months*`]` indicate that there are additional optional parameters that may be added to the `readership` command. For example, you could use `readership` command with the `-d` option to count the number of people who have read messages in a shared folder within the indicated number of days:

```
readership -d 10
```

Command Line Prompts

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

Platform-specific Syntax

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

Related Third-Party Web Site References

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 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/coll/MessagingServer_04q2`

Listed below are the additional documents that are available:

- *Sun Java System Messaging ServerRelease Notes*
- *Sun Java System Messaging Server Administration Guide*
- *Sun Java System Messaging Server Developer's Reference*
- *Sun Java System Messaging Server Deployment Planning Guide*
- *Sun Java System Messenger Express Customization Guide*
- *Sun Java System Communications Services Enterprise Deployment Planning Guide*
- *Sun Java System Communications Services Event Notification Service Guide*
- *Sun Java System Communications Services Schema Migration Guide*
- *Sun Java System Communications Services Schema Reference*
- *Sun Java System Communications Services User Management Utility Administration Guide*
- *Sun Java System Communications Express Customization Guide*
- *Sun Java System Communications Express Administration Guide*

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/coll/MessagingServer_04q2

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

Where to Find This Book Online

You can find the *Messaging Server Administration Reference* online in PDF and HTML formats. To find this book, use this URL:

http://docs.sun.com/coll/MessagingServer_04q2

Chapter 1

Messaging Server Command-line Utilities

Sun Java System Messaging Server provides a set of command-line utilities in addition to its graphical user interface. This chapter describes utilities for messaging server starting, stopping, administration, message access, and message store.

For descriptions of the command-line utilities for the MTA, see Chapter 2, "Message Transfer Agent Command-line Utilities".

The commands described in this chapter are listed in Table 1-1.

Table 1-1 Messaging Server Commands

Command	Description
configutil	Enables you to list and change Messaging Server configuration parameters.
counterutil	Displays all counters in a counter object. Monitors a counter object.
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 account.
imexpire	Expires and purges messages in the Message Store
iminitquota	Reinitializes the quota limit from the LDAP directory and recalculates the disk space being used.
immonitor-access	Monitors the status of the Messaging Server components.
imquotacheck	Calculates the total mailbox size for each user in the message store and compares the size with their assigned quota.
imsasm	Handles the saving and recovering of user mailboxes.
imsbackup	Backs up stored messages.

Table 1-1 Messaging Server Commands *(Continued)*

Command	Description
imsconnutil	Monitors user access of the message store.
imsexport	Exports Sun Java System Messaging Server mailboxes into UNIX `/var/mail` format folders.
imsimport	Migrates UINX `/var/mail` format folders into an Sun Java System Messaging Server message store.
imsrestore	Restores messages from the backup device into the message store.
imscripter	The IMAP server protocol scripting tool. Executes a command or sequence of commands.
mboxutil	Lists, creates, deletes, renames, or moves mailboxes (folders).
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.
msuserpurge	Purges those user and domain mailboxes from the message store.
readership	Reports on how many users other than the mailbox owner have read messages in a shared IMAP folder.
reconstruct	Rebuilds one or more mailboxes, or the master mailbox file, and repairs any inconsistencies.
refresh	Refreshes the configuration of the specified messaging server processes
start-msg	Starts the messaging server processes.
stop-msg	Stops the messaging server processes.
stored	Performs cleanup and expiration operations.

Command Descriptions

This section describes what the main Sun Java System Messaging Server command-line utilities do, defines their syntax, and provides examples of how they are used. The utilities are listed in alphabetical order.

Store programs do a `setuid` at configuration initialization time if running as `root`. IMAPD and POP3 do a `setuid` after opening the sockets, since that needs root privilege. The few files (logs, locks) which may be created before the uid change are chowned to mailsrv so that they are still usable by utilities starting directly as mailsrv.

configutil

The `configutil` utility enables you to list and change Sun Java System Messaging Server configuration parameters.

For a list of all configuration parameters, see Chapter 3, "Messaging Server Configuration."

Most Sun Java System Messaging Server configuration parameters and values are stored in the LDAP database on Directory Server with the remaining parameters and values stored locally in the `msg.conf` and `local.conf` files. The startup parameters are stored in the `msg.conf` file and are set during installation. The `local.conf` files should not be edited manually. Use configutil to edit the parameters stored in those files.

NOTE If the administrator has defined any language-specific options (such as messages), you must use the `language` option at the end of the command in order to list or change them. Commands entered without a `language` option are only applied to attributes that do not have a specified language parameter.

Requirements: Must be run locally on the Messaging Server. You may run `configutil` as `root` or `mailsrv`. If you make changes to the servers, you must restart or refresh the servers, depending on the variable, for the changes to take effect.

Location: *msg_svr_base*`/sbin/configutil`

You can use `configutil` to perform four tasks:

- Display particular configuration parameters using `-o` *option*.
  - Add `;lang-`*xx* after the option to list parameters with a specified language parameter. For example, `;lang-jp` to list options specified for the Japanese language.
- List configuration parameter values using the `-l` or `-p` *prefix* options.
  - Use `-l` to just list local configuration parameters from the server's local configuration file.
  - Use `-p` *prefix* to just list those configuration parameters whose names begin with the letters specified in *prefix*.
- Set configuration parameters using the `-o` *option* and `-v` *value* options.

- Include the `-l` option with `-o` *option* and `-v` *value* to store the new value in the server's local configuration file.
- To read the actual value from `stdin`, specify a dash (`-`) as the *value* on the command line.
- Add `;lang-`*xx* after the option to set options for a specified language parameter. For example, `;lang-jp` to set options specified for the Japanese language.

• Import configuration parameter values from `stdin` using the `-i` option.

- Include the `-l` option with the `-i` option to import all configuration parameters to the server's local configuration file.

Syntax

```
configutil [-f configdbfile] [-o option [;language] [-v value]]

configutil [-f configdbfile] [-p prefix][;language]

configutil [-f configdbfile] -l[-o option [;language] [-v value]]

configutil -i inputfile
```

Options

The options for this command are:

Option	Description
`-f` *configdbfile*	Enables you to specify a local configuration file other than the default. (This option uses information stored in the `CONFIGROOT` environment variable by default.)
`-i` *inputfile*	Imports configurations from a file. Data in the file to be entered in *option*|*value* format with no spaces on either side of the pipe. The *inputfile* should be specified as an absolute path.
`-l`	Lists configuration parameters stored in the local server configuration file. When used in conjunction with the `-v` option, specifies that a configuration parameter value be stored in the local server configuration file.

Option	Description
`-o` *option*	Specifies the name of the configuration parameter that you wish to view or modify. May be used with the `-l` and `-i` options. Configuration parameter names starting with the word `local` are stored in the local server configuration file.
`-p` *prefix*	Lists configuration parameters with the specified prefix.
`-v` *value*	Specifies a value for a configuration parameter. To be used with `-o` *option*. If the `-l` option is also specified or the configuration parameter name specified with the `-o` option begins with `local`, the option value is automatically stored in the local server configuration file rather than the Directory Server.

If you specify no command-line options, all configuration parameters are listed.

Examples

To list all configuration parameter and their values in the both the Directory Server LDAP database and local server configuration file:

```
configutil
```

To import configurations from an input file named `config.cfg`:

```
configutil -i config.cfg
```

To list all configuration parameters with the prefix `service.imap`:

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

To display the value of the `service.smtp.port` configuration parameter:

```
configutil -o service.smtp.port
```

To set the value of the `service.smtp.port` configuration parameter to 25:

```
configutil -o service.smtp.port -v 25
```

To clear the value for the `service.imap.banner` configuration parameter:

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

Language Specific Options

To list or set options for a specific language, append `;lang-`*xx* immediately after the option with no spaces, where *xx* is the two-letter language identifier. For example, to view the text of the Japanese version of the `store.quotaexceededmsg` message:

```
configutil -o "store.quotaexceededmsg;lang-jp"
```

counterutil

The `counterutil` utility displays and changes counters in a counter object. It can also be used to monitor a counter object every 5 seconds.

Requirements: Must be run locally on the Messaging Server as root.

Location: *msg_svr_base*`/sbin/`

Syntax

```
counterutil -o counterobject [-i interval] [-l] [-n numiterations]
  [-r registryname]
```

Options

The options for this command are:

Option	Description
`-i` *interval*	Specifies, in seconds, the interval between reports. The default is 5.
`-l`	Lists the available counter objects in the registry specified by the `-r` option.
`-n` *numiterations*	Specifies the number of iterations. The default is infinity.
`-o` *counterobject*	Continuously display the contents of a particular counter object every 5 seconds.
`-r` *registryname*	Indicates the counter registry to use. If no *registryname* is specified with the `-r` *registryname* option, the default is *msg_svr_base*`/counter/counter`.

Examples

To list all counter objects in a given server's counter registry:

```
counterutil -l
```

To display the content of a counter object imapstat every 5 seconds:

```
counterutil -o imapstat -r \
msg_svr_base/counter/counter
```

deliver

The `deliver` utility delivers mail directly to the message store accessible by IMAP or POP mail clients.

If you are administering an integrated messaging environment, you can use this utility to deliver mail from another MTA, a `sendmail` MTA for example, to the Messaging Server message store.

NOTE The `deliver` utility is only for use with files which are already completely and properly formed email messages.

Requirements: Must be run locally on the Messaging Server; the `stored` utility must also be running. Make sure that the environment variable CONFIGROOT is set to *msg_svr_base*`/config`.

Location on UNIX: *msg_svr_base*`/sbin/`

Syntax

```
deliver [-l] [-c] [-d] [-r address] [-f address] [-m mailbox]
  [-q] [-g flag] [userid]...
```

You can specify multiple userids.

Options

The options for this command are:

Option	Description
`-a` *authid*	Specifies the authorization ID of the sender. Defaults to anonymous.
`-c`	Automatically creates the mailbox if it doesn't exist in the message store.
`-d`	This option is recognized by `deliver` in order to maintain compatibility with `/bin/mail`, but it is ignored by `deliver`.
`-g` *flag*	Sets the system flag or keyword flag on the delivered message.
`-f` *address*	Inserts a forwarding path header containing address.
`-l`	Accepts messages using the LMTP protocol (RFC 2033).
`-m` *mailbox*	Delivers mail to *mailbox*. • If any user ids are specified, attempts to deliver mail to *mailbox* for each user id. If the access control on a mailbox does not grant the sender the "p" right or if the `-m` option is not specified, then this option delivers mail to the inbox for the user ID, regardless of the access control on the inbox. • If no user ids are specified, this option attempts to deliver mail to *mailbox*. If the access control on a mailbox does not grant the sender the "p" right, the delivery fails.
`-q`	Overrides mailbox quotas. Delivers messages even when the receiving mailbox is over quota.
`-r` *address*	Inserts a `Return-Path:` header containing address.

Option	Description
userid	Deliver to inbox the user specified by *userid*.

If you specify no options, mail is delivered to the inbox.

Examples

To deliver the contents of a file named `message.list` to Fred's `tasks` mailbox:

```
deliver -m tasks fred < message.list
```

In the above example, if the `tasks` mailbox does not grant "p" rights to the sender, the contents of `message.list` are delivered to the inbox of the user `fred`.

hashdir

The `hashdir` command identifies the directory that contains the message store for a particular account. This utility reports the relative path to the message store. The path is relative to the directory level just before the one based on the user ID. `hashdir` sends the path information to standard output.

Requirements: Must be run locally on the Messaging Server. Make sure that the environment variable CONFIGROOT is set to *msg_svr_base*`/config`.

Location: *msg_svr_base*`/sbin/`

Syntax

```
hashdir [-a] [-i] account_name
```

Options

The options for this command are:

Option	Description
`-a`	Appends the directory name to the output.

Option	Description
-i	Allows you to use the command in interactive mode.

Examples

```
hashdir user1
```

imexpire

imexpire automatically removes messages from the message store based on administrator-specified criteria. The criteria can be set in the Admin Console GUI, with configutil parameters, or in a file called store.expirerule. (See the *Sun Java System Messaging Server Administration Guide* detailed usage information.) The following removal criteria can be specified:

- Folder pattern
- Number of messages in the mailbox
- Total size of the mailbox
- Age, in days, that messages have been in the mailbox
- Size of message and grace period (days that a message exceeding the size limit will remain in the message store before removal)
- Whether a message has been flagged as *seen* or *deleted*
- By header and field

NOTE The functionality of imexpire has been expanded and the interface has changed since earlier versions of Messaging Server. However, this version continues to support older imexpire configurations.

Requirements: Must run on local machine (the machine that holds the message store files). Some or all of the following may be required: local.schedule.expire, local.schedule.purge, store.cleanupage, local.store.expire.loglevel, store.expirerule.*rulename.attribute*, store.expirestart, local.store.expire.workday, local.store.expire.cleanonly

Location: *msg_svr_base*/sbin

Syntax

```
imexpire [-c] [-e] [-n] [-v] [-d] [-p partition]
```

Options

The options for this command are:

Option	Description
-c	Do purge only—do not expire. Remove expunged and expired messages.
-e	Do expire only - do not purge.
-n	Trial run only - do not perform expire or cleanup. A description of what would happen without this flag is output.
-v 1|2|3	Display verbose output. The number specifies the loglevel, where 1= partition level 2 = mailbox level 3 = message level Messages are logged to the log file by default. When the -d option is used, messages go to stderr.
-d	Display debug output to stderr.
-p *message_store_partition*	Expire/Purge the message store partition specified.

Examples

Purge expunged messages with verbose output.

```
imexpire -c -v
```

iminitquota

The `iminitquota` utility reinitializes the quota limit from the LDAP directory and recalculates the total amount of disk space that is being used by the users. It updates the message store `quota.db` database under the `mboxlist` directory in the message store. The `iminitquota` utility should be run after the `reconstruct -q` utility is run.

Location: *msg_svr_base*`/sbin/`

Syntax

```
iminitquota -a | -u userid
```

Options

The options for this command are:

Option	Description
`-a`	Initializes and updates the quota files for every message store user.
`-u` *userid*	Reinitializes and updates the quota-related information for the specified user. The *userid* parameter specifies the message store id of a user, not the login id of the user.

You must specify either the `-a` or `-u` option with the `iminitquota` command.

immonitor-access

Monitors the status of Messaging Server components—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.

The internal operation of `immonitor-access` is as follows: first it does an `ldapsearch` of a test user created by the administrator. This checks the Directory Server. It can then connect to the SMTP port and send a message to the mail address to check the dispatcher. Then, it checks Message Access by using the IMAP and POP server to see if the message made it to the Message Store. The command logs a message in the default log file if any of the thresholds are exceeded.

The command creates a report that contains the following information:

- The state of the components
- The response time
- The round-trip time for that service

`immonitor-access` is typically run by `cron` at scheduled intervals to provide a snapshot of the status of the Message Access and Store components. `immonitor-access` can also connect to the IMAP / POP service and delete messages with the subject specified by `-k`. If `-k` is not specified, all messages containing the subject header, `immonitor`, are deleted.

The administrator must create a test user for use by this command before it can be executed.

Syntax

`immonitor-access -u` *user_name* { `[-L` *LDAP_host*`:[`*port*`]=[`*threshold*`] ] [-b` *searchbase*`] [-I` *IMAP_host*`:[`*port*`]=[`*threshold*`] ] [-P` *POP_host*`:[`*port*`]=[`*threshold*`] ] [-H` *HTTP_host*`:[`*port*`]=[`*threshold*`] ] [-S` *SMTP_host*`:[`*port*`]=[`*threshold*`] ] [-w` *passwd*`] } [-D` *threshold*`] [-m` *file*`] [-r` *alert_recipients*`] [-A` *Host*`] [-f` *SOMSADMIN*`] [-C` *LMTP_host*`:[`*port*`]=[`*threshold*`] ] [-hdv]`

`immonitor access -u` *user_name* `-w` *passwd* `{ [ -I` *IMAP_host*`:[`*port*`]=[` *threshold* `]] [ -P` *POP_host*`: [`*port*`]=[` *threshold* `]]} [-k` *subject*`] -z`

`immonitor access -u` *user_name* `-w` *passwd* `{ [ -H` *HTTP_host*`:[`*port*`]=[` *threshold* `]] }`

Options

The following list contains valid task options for the command.

Option	Description
`-u` *user_name*	The valid test user account to use. This test mail user has to be created by the administrator. If the test mail user is in a hosted domain, *user@domain* should be specified.

Option	Description
-w *passwd*	The password corresponding to the user specified with -u. This option is mandatory when the -I or -P is used. "-" can specified with -w, to enter the password through standard input.
-L *LDAP_host*: [*port*] = [*threshold*]	Use the LDAP server and the port specified to check the Directory Server. The threshold is specified in seconds.
-I *IMAP_host*: [*port*] = [*threshold*]	Use the IMAP server and the port specified to check the IMAP component of the Message Access. The threshold is specified in seconds. The threshold involves the time to login, retrieve, and delete the message.
-P *POP_host*: [*port*] = [*threshold*]	Use the POP server and the port specified to check the POP component of the Message Access. The threshold is specified in seconds. The threshold involves the time to login, retrieve, and delete the message.
-S *SMTP_host*: [*port*] = [*threshold*]	Use the SMTP server and the port specified to check if Messaging Server is able to accept mail for delivery. The threshold is specified in seconds.
-C *LMTP_host*: [*port*] = [*threshold*]	Use the LMTP server and the port specified to check if Messaging Server is able to deliver the message to the store. The threshold is specified in seconds.
-H *HTTP_host*: [*port*] = [*threshold*]	Use the HTTP server and the port specified to check if the HTTP server is able to accept requests on the specified port. When -I -H or -P is used, it is necessary to provide the test user password with -w. The default ports are: SMTP = 25 IMAP = 143 POP = 110 LDAP = 389 LMTP = 225 HTTP = 80. If either the port or threshold is not specified, default ports with the default threshold of 60 seconds is assumed. The threshold specified can be a decimal number. When -S/-C, -I/-P are specified together, the command does the following: - sends mail and retrieves with IMAP and POP - reports the per protocol response time - reports round-trip time o reports delivery time (the time taken to send the mail and be visible to IMAP/POP) Multiple -I, -P, and -S options can be specified, which helps in monitoring Messaging Server on various systems.

Option	Description
`-b` *searchbase*	Use search base as the starting point for the searching in the Directory Server. It is the same as `-b` of `ldap-search(1)`. If `-b` is not specified, the utility uses the value of dcRoot of the configuration parameter `local.ugldapbasedn`.
`-f` *mail From option:*	When `immonitor-access` sends out an e-mail, it usually is sent as root@*domainname*. Specify this option to send out an e-mail as different user: `-f user@red.iplanet.com`
`-D` *threshold*	The delivery (also called round-trip time) threshold. The time taken to send the mail and the mail being visible to POP/IMAP. This option can be used only when `-I`/`-P` and `-S`/`-C` are used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Read the command information from a file instead of from the command line.
`-m` *file*	The file that is mailed to the test user. You can get response and round-trip times for various mail sizes with this option. Specify only text files as non-text files result in unexpected behavior. If `-m` is not specified, the `mailfile.txt` file in *msg_svr_base*`/lib/locale/C/mailfile.txt` is used as the mail file.
`-k` *subject*	Header subject of the messages to be sent/deleted. The utility, by default, uses the string “immonitor:<date>” as the subject in the header sent out with the `-S` option. If `-k` is specified, the string “immonitor:subject” is used in the subject header. This option can be used with `-z` to delete messages, if `-k` is not specified, all messages with the Subject header containing “immonitor” are deleted.
`-z`	Delete messages containing the string specified by `-k` in the subject header. If `-k` is not specified, all messages with the subject header containing “immonitor” are deleted. Use `-z` only with `-I` or `-P`. Do note use `-z` with `-S` or `-C` as this can cause unexpected results.
`-r` *alert_recipients*	A comma-separated list of mail recipients who will be notified. If this option is not specified, the command reports the alert messages on the standard output.
`-A` *host*	The alternate mail server to be used to send mail to the *alert_recipients*. This option helps in sending alert messages even when the primary mail server is down or heavily loaded. If `-A` is not specified, the SMTP server on the localhost is used.
`-h`	Display the usage message.
`-d`	The debug mode: display the execution steps.
`-v`	Run in verbose mode, with diagnostics written to standard output.

Output

The command generates a report containing the various protocol execution times. For example:

```
Smtp Statistics for: thestork:25
Connect Time: 2.122 ms
Greeting Time: 5.729 ms
Helo Time: 2.420 ms
Mail From: Time: 2.779 ms
Rcpt To: Time: 4.128 ms
Data Time: 1.268 ms
Sending File Time: 94.156 ms
Quit Time: 0.886 ms
Total SMTP Time: 113.488 Milliseconds
```

If the alert recipients are specified and any of the threshold values are exceeded, the command mails the report containing the service name and the response time:

```
ALERT: <service> exceeds threshold
Response time=secs/Threshold=secs
```

Note that in case of times reported for IMAP, the individual times might not add up to the exact value shown by the “Total IMAP time”. This occurs because the message does not get to the store immediately. The utility loops until the message is found. Typically, the search time indicates only the successful search time. However, the total time includes each of the individual sleep and search times.

With POP, the utility needs to login and logout multiple times before the message is actually found in the store. Thus, the total time here is the accumulated time for all the logins and log outs.

Examples: To monitor the LDAP, SMTP, IMAP and POP with the threshold of 10 seconds and 250 milliseconds on localhost use:

```
immonitor access -L localhost:=60.25   -S \
localhost:=60.25 -I localhost:=60.25 -P localhost:=60.25 \
-u test_user -w passwd
```

This example assumes that `test_user` exists with password “`passwd`”

Exit Status

The exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error. A different exit status is returned when various thresholds are exceeded.

0 Successful execution with no errors or thresholds exceeded

1 Exceeded threshold of a service

2 Errors

64 Usage errors

An alert message is written to the console when the response time of any server exceeds the threshold.

An error message is written to the console when any of the servers cannot be reached.

Warnings

The password passed with `-w` can be visible to a user using the `ps(1)` command. It is strongly advised that you create a test user to be specifically used by the monitoring utilities.

It is recommended that you use `-w` and enter the password through standard input. However, if the utility is executed through `cron`, the password can be stored in a file. This file can be redirected as the standard input for the utility.

```
cat passwd_file | immonitor -w -
immonitor -w - ... < passwd_file
```

Do not use the echo command such as:

```
echo password | immonitor .. -w - ..
```

because the `ps` might show the echo's arguments.

To delete the test mail sent by the `-S` option, invoke the `immonitor access` command with the `-z` option separately. Do not use the two together.

imquotacheck

The `imquotacheck` utility calculates the total mailbox size for each user in the message store. 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.

Requirements: Must be run locally on the Messaging Server.

Dependencies: The delivery agent's quota warning mechanism needs to be turned off in order for `imquotacheck` to work, because the `imquotacheck` and the delivery agent use the same element in the quota database to record last-warn time. To turn off the delivery agent's quota warning, set `store.quotanotification` to `off`.

Location: *msg_svr_base*`/sbin/`

Syntax

The following form of `imquotacheck` should be used when you want to notify users if they have exceeded a set percentage of their assigned quota.

```
imquotacheck [-e] [-d domain][-f] [-r rulefile] [-t message template] [-D] -n
```

This following form of `imquotacheck` should be used when you want to report the usage to `stdout`.

```
imquotacheck [-e] [-d domain] [-r rulefile] [-t message template] [-i] [-v]
  [-h] [-u user] [-D]
```

Options

The options for this command are:

Option	Description
`-e`	Allows extended reporting. Per folder usage is included in the report.
`-d` *domain*	Looks for users only in the specified domain. The `-i` option is implied, so it does not need to be specified.
`-f`	Enforces domain quotas. If the domain is over quota and the `maildomainstatus` attribute is currently set to active, the value will be reset to overquota, which will prevent mail from being accepted by the message store. If the domain is not over quota and the `maildomainstatus` attribute is set to overquota, then the value will be changed to active, and mail will be accepted.
`-r` *rulefile*	Specifies the set of rules to be used when you want to calculate quota usage. If `-r` is not specified, a default *rulefile* can be used. To setup a default *rulefile*, copy the "Sample Rulefile" on page 38 to *msg_svr_base*`/config`. See "Rulefile Format" on page 34.

Option	Description
`-t` *message template*	Notifies users when their mailbox quota is exceeded. The message template format is the following: • %U% - user's mailbox id • %Q% - percentage of the used mailbox quota • %R% - quota usage details: assigned quota, total mailbox size, and percentage used. If the -e is specified, mailbox usage of the individual folders are also reported. • %M% - current mailbox size • %C% - quota attribute value If -t is not specified, a default message file will be mailed. To setup a default message file, copy the "Notification File" on page 39 to *msg_svr_base*`/config`.
`-n`	Sends notification messages based on the rules defined in the *rulefile*. If you do not define any rules when you use this option, you will receive an error.
`-i`	Ignores the *rulefile* and any active rule defined in it. The quota status of all the users in the message store will be printed to `stdout`. This option can only be used when you want to report usage. If -i is not specified, the active rule with the least threshold is used to print a list of all of the users and their quota status to `stdout`.
`-v`	Prints the username, quota, total mailbox size and percentage of mailbox used by all of the users. When you are using `imquotacheck` to report usage, it will default to this option if no other options are specified.
`-u` *user*	Obtains the quota status of the specified user id. Can be used with `-e` for extended reporting on the user. Cannot be used to specify multiple users.
`-D`	Debug mode; displays the execution steps to `stdout`.

Examples

To send a notification to all users in accordance to the default `rulefile`:

```
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`:

```
imquotacheck -n -r myrulefile -t mytemplate.file
```

To list the usage of all users whose quota exceeds the least threshold in the `rulefile`:

```
imquotacheck
```

To list the usage of all users and (will ignore the `rulefile`):

```
imquotacheck -i
```

To list per folder usages for users `user1` (will ignore the `rulefile`):

```
imquotacheck -u user1 -e
```

To only list the users in domain `siroe.com`:

```
imquotacheck -d siroe.com
```

Rulefile Format

The `rulefile` format is organized into two sections: a general section and a rule name section. The general section contains attributes that are common across all rules. Attributes that are typically specified in the general section are the `mailQuotaAttribute` and the `reportMethod`. In the rule name section, you can write specific quota rules for notification intervals, trigger percentages, and so on. Attributes that are typically specified in the rule name section are `notificationTriggerPercentage`, `enabled`, `notificationInterval`, and `messageFile`. Note that the attributes and corresponding values are not case-sensitive. The following rulefile format is used:

```
[General]
mailQuotaAttribute = [value]
reportMethod = [value]

[rulename1]
attrname=[value]
attrname=[value]

[rulename2]
attrname=[value]
attrname=[value]

[rulename3]
attrname=[value]
attrname=[value
```

This table shows the attributes, whether they are required, the default value, and the description.

General Attribute	Required Attribute?	Default Value	Description
`mailQuotaAttribute`	No	Value in `quotadb`	Specifies the name of the custom mailquota attribute. If not specified, the value in `quotadb` is used.
`reportMethod`	No		Can customize the output of the quota report. The value of this attribute is specified as *library-path*:*function*, where *library-path* is the path of the shared library and *function* is the name of the report function. See "`reportMethod` Signature" on page 36 to see the structure of the attribute.

Rule Attribute	Required Attribute ?	Default Value	Description
`notificationTriggerPercentage`	Yes		Specifies the consumed quota percentage that will trigger notification. Value should be unique and an integer.
`messageFile`	No	*msg_svr_base*`/config/imq.msgfile`	Specifies the absolute path to the message file.

Rule Attribute	Required Attribute ?	Default Value	Description
`notificationInterval`	Yes		Indicates the number of hours before a new notification is generated.
`enabled`	No	`0` (FALSE)	Indicates if the particular rule is active. Applicable values are `0` for FALSE and `1` for TRUE.
`notificationMethod`	No		Can customize the overquota notification method to send to the user. The value of this attribute is specified as *library-path:function*, where l*ibrary-path* is the path of the shared library and *function* is the name of the report function. See "`notificationMethod` Signature" on page 37 to see the structure of the attribute.

`reportMethod` Signature

The following signature can be used for the `reportMethod()`:

```
int symbol(QuotaInfo* info, char** message, int* freeflag)
info is a pointer to the following structure:
typedef struct QuotaInfo {
  const char* username; /* user name (uid or uid@domain) */
  long quotakb; /* quota in kbytes */
  long quotamsg;  /* quota in number of messages */
  ulong usagekb; /* total usage in kbytes */
  ulong usagemsg; /* total usage in number of messages */
  FolderUsage* folderlist; /* folder list (for -e) */
  long num_folder;  /* number of folders in the folderlist */
  long trigger; /* not used */
  const char* rule; /* not used */
}

typedef struct FolderUsage {
  const char*foldername;
  ulong usagekb; /* folder usage in kbytes */
}
```

The address, message, points to the output message. The report function is expected to fill the value of *message and allocate memory for message when necessary. The freeflag variable indicates if the caller is responsible for freeing allocated memory for *message.

The return values are 0 for success and 1 for failure.

The imquotacheck function will invoke the reportMethod to generate the report output. If the reportMethod returns 0 and *message is pointing to a valid memory address, message will be printed.

If the *freeflag is set to 1, the caller will free the memory address pointed to by message. If the -e option is specified, the quota usage for every folder will be stored in the folderlist, an array in FolderUsage; the num_folder variable is set to the number of folders in the folderlist.

notificationMethod Signature

The following signature can be used for the notificationMethod():

```
The notification function has the following prototype:
int symbol(QuotaInfo* info, char** message, int* freeflag)
info is a pointer to the following structure:
typedef struct QuotaInfo {
  const char* username; /* user name (uid or uid@domain) */
  long quotakb; /* quota in kbytes */
  long quotamsg;  /* quota in number of messages */
  ulong usagekb; /* total usage in kbytes */
  ulong usagemsg; /* total usage in number of messages */
  FolderUsage* folderlist; /* folder list (for -e) */
  long num_folder;  /* number of folders in the folderlist */
  long trigger; /* the exceeded notificationTriggerPercentage */
  const char* rule; /* rulename that triggered notification */
}

typedef struct FolderUsage {
  const char *foldername;
  ulong usagekb; /* folder usage in kbytes */
}
```

The address, message, points to the notification message. The notification function is expected to fill in the value of this variable and allocate the memory for the message when necessary. The freeflag variable indicates if the caller is responsible for freeing the memory allocated for message.

The return values are 0 for success and 1 for failure.

If the notification function returns a 0, and `*message` is pointing to a valid address, the `imquotacheck` utility will deliver the message to the user. If the `*freeflag` is set to 1, the caller will free the memory address pointed to by message after the message is sent.

If the `-e` option is specified, the quota usage for every folder will be stored in the `folderlist` variable, an array of `FolderUsage` structure; the `num_folder` variable is set to the number of folders in the `folderlist`.

NOTE	If the `messageFile` attribute is also specified, the attributed `messageFile` will be ignored.

Sample Rulefile

```
#
# Sample rulefile
#
[General]
mailQuotaAttribute=mailquota
reportMethod=/xx/yy/libzz.so:myReportMethod [for Solaris only ]
             /xx/yy/libzz.sl:myReportMethod [for HP-UX only]
             \xx\yy\ibzz.dll:myReportMethod [for Windows NT only]

[rule1]
notificationTriggerPercentage=60
enabled=1
notificationInterval=3
notificationMethod=/xx/yy/libzz.so:myNotifyMethod_60

[rule2]
notificationTriggerPercentage=80
enabled=1
notificationInterval=2
messageFile=/xx/yy/message.txt

[rule3]
notificationTriggerPercentage=90
enabled=1
notificationInterval=1
notificationMethod=/xx/yy/libzz.so:myNotifyMethod_90

#
# End
#
```

Threshold Notification Algorithm

1. Rule precedence is determined by increasing trigger percentages.
2. The highest applicable threshold is used to generate a notification. The time and the rule's threshold are recorded.
3. If users move into a higher threshold since their last quota notification, a new notification will be delivered based on the current set of applicable rules. This notice can be immediately delivered to any user whose space usage is steadily increasing.
4. If usage drops, the notification interval of the current rule (lower threshold) will be used to check the time elapsed since the last notice.
5. The stored time and threshold for the user will be reset to zero if the user's mailbox size falls below all of the defined thresholds.

Notification File

The utility depends on the message file to have at minimum a Subject Header. There should be at least one blank line separating the Subject from the body. The other requires headers are generated by the utility/ The notification file format is the following:

```
Subject: [Warning] quota reached for %U%


Hello %U%,
Your quota: %C%
Your current mailbox usage: %M%
Your mailbox is now %Q% full. The folders consuming the most space are:
%R%.


Please clean up unwanted diskspace.

Thanks,
-Administrator
```

NOTE 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.

imsasm

The `imsasm` utility is an external ASM (Application Specific Module) that handles the saving and recovering of user mailboxes. `imsasm` invokes the `imsbackup` and `imsrestore` utilities to create and interpret a data stream.

During a save operation `imsasm` creates a save record for each mailbox or folder in its argument list. The data associated with each file or directory is generated by running the `imsbackup` or `imsrestore` command on the user's mailbox.

Location: *msg_svr_base*`/lib/msg`

Syntax

```
imsasm [standard_asm_arguments]
```

Options

The options used in the `imsasm` utility are also known as standard-asm-arguments, which are Legato NetWorker® backup standards.

Either `-s` (saving), `-r` (recovering), or `-c` (comparing) must be specified and must precede any other options. When saving, at least one *path* argument must be specified. *path* may be either a directory or filename.

The following options are valid for all modes:

Option	Description
`-n`	Performs a dry run. When saving, walk the file system but don't attempt to open files and produce the save stream. When recovering or comparing, consume the input save stream and do basic sanity checks, but do not actually create any directories or files when recovering or do the work of comparing the actual file data.
`-v`	Turns on verbose mode. The current ASM, its arguments, and the file it is processing are displayed. When a filtering ASM operating in filtering mode (that is, processing another ASM's save stream) modifies the stream, its name, arguments, and the current file are displayed within square brackets.

When saving (`-s`), the following options may also be used:

Option	Description
`-b`	Produces a byte count. This option is like the `-n` option, but byte count mode will estimate the amount of data that would be produced instead of actually reading file data so it is faster but less accurate than the `-n` option. Byte count mode produces three numbers: the number of records, i.e., files and directories; the number of bytes of header information; and the approximate number of bytes of file data. Byte count mode does not produce a save stream so its output cannot be used as input to another asm in recover mode.
`-o`	Produces an “old style” save stream that can be handled by older NetWorker servers.
`-e`	Do not generate the final “end of save stream” Boolean. This flag should only be used when an ASM invokes an external ASM and as an optimization chooses not to consume the generated save stream itself.
`-i`	Ignores all save directives from `.nsr` directive files found in the directory tree.
`-f` *proto*	Specifies the location of a `.nsr` directive file to interpret before processing any files. Within the directive file specified by *proto*, *path* directives must resolve to files within the directory tree being processed, otherwise their subsequent directives will be ignored.
`-p` *ppath*	Prepends this string to each file’s name as it is output. This argument is used internally when one ASM executes another external ASM. *ppath* must be a properly formatted path which is either the current working directory or a trailing component of the current working directory.
`-t` *date*	The date after which files must have been modified before they will be saved.
`-x`	Crosses file system boundaries. Normally, file system boundaries are not crossed during walking.

When recovering (`-r`), the following options may also be used:

Option	Description
`-i` *response*	Specifies the initial default overwrite response. Only one letter may be used. When the name of the file being recovered conflicts with an existing file, the user is prompted for overwrite permission. The default response, selected by pressing `Return`, is displayed within square brackets. Unless otherwise specified with the `-i` option, `n` is the initial default overwrite response. Each time a response other than the default is selected, the new response becomes the default. When either `N`, `R`, or `Y` is specified, no prompting is done (except when auto-renaming files that already end with the rename suffix) and each subsequent conflict is resolved as if the corresponding lower case letter had been selected. The valid overwrite responses and their meanings are: • `n`—Do not recover the current file. • `N`—Do not recover any files with conflicting names. • `y`—Overwrite the existing file with the recovered file. • `Y`—Overwrite files with conflicting names. • `r`—Rename the conflicting file. A dot "." and a suffix are appended to the recovered file's name. If a conflict still exists, the user will be prompted again. • `R`—Automatically renames conflicting files by appending a dot "." and a suffix. If a conflicting file name already ends in a `.`*suffix*, the user will be prompted to avoid potential auto rename looping conditions.
`-m` *src*=*dst*	Maps the file names that will be created. Any files that start exactly with src will be mapped to have the path of dst replacing the leading src component of the path name. This option is useful if you wish to perform relocation of the recovered files that were saved using absolute path names into an alternate directory.
`-z` *suffix*	Specifies the suffix to append when renaming conflicting files. The default suffix is R.
path	Restricts the files being recovered. Only files with prefixes matching path will be recovered. This checking is performed before any potential name mapping is done with the -m option. When path is not specified, no checking is performed.

Examples

To use imsasm to save the mailbox INBOX for user joe, the system administrator creates a directory file *backup_root*/backup/DEFAULT/joe/.nsr with the following contents:

```
imsasm: INBOX
```

This causes the mailbox to be saved using imsasm. Executing the mkbackupdir utility will automatically create the .nsr file. See "mkbackupdir" on page 50.

imsbackup

The imsbackup utility is used to 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 via the imsrestore utility. The imsbackup utility provides a basic backup facility similar to the UNIX tar command.

Location: *msg_svr_base*/sbin

For more information about imsbackup and backing up the message store, see the section "Backing Up and Restoring the Message Store" in the *Messaging Server Administration Guide*.

Syntax

```
imsbackup -f device [-b blocking_factor] [-d datetime] [-e encoding] [-u file]
[-m linkcount] [-ivlgx] [name ...]
```

Options

The options for this command are:

Option	Description
-b *blocking_factor*	Everything written to the backup device is performed by blocks of the size 512x*blocking_factor*. The default is 20.

Option	Description
`-d` *datetime*	Date from which messages are to be backed up, expressed in *yyyymmdd*[*:hhmmss*]; for example, `-d 19990501:13100` would backup messages stored from May 1, 1999 at 1:10 pm to the present. The default is to backup all the messages regardless of their dates.
`-e` *encoding*	Mailbox name encoding (example: IMAP_MODIFIED-UTF-7)
`-f` *device*	Specifies the file name or device to which the backup is written. If `device` is '-', backup data is written to `stdout`.
`-g`	Debug mode
`-i`	Ignore links. Used for partial store.
`-l`	Used to autoload tape devices when end-of-tape is reached.
`-m` *link_count*	Specifies the minimum link count for hashing.
`-u` *file*	Specifies a backup object file. This file contains the object names (entire message store, user, group, mailbox, and so on) to restore. See *name* for a list of backup object
`-v`	Executes the command in verbose mode.

Option	Description
name	Can be 1) logical pathname of the backup object, 2) user ID, 3) message store mailbox name. Backup objects and paths: • Entire message store: `/` • Message store partition: /*partition_name* (default: `/primary`) • Backup group—a group of users defined with regular expressions in a configuration file. See *backup groups* in the *Sun Java System Messaging Server Administration Guide* for details. Path: /*partition_name/backup_group* (`/primary/user` represents all users under `primary`). • User: /*partition_name/backup_group/user_ID* • Mailbox: /*partition_name/backup_group/user_ID/mailbox_name* • Message: /*partition_name/backup_group/user_ID/mailbox_name/msgID* User IDs: can be any user ID in the message store. If the user is not in the default domain, the user ID must be fully qualified (example: Wally@siroe.com). If user is in the default domain, the user ID can stand alone (example: Wally). *mailbox:* An email folder. It is specified using the following message store internal name: `user`/*user_ID*/*folder_name*. Note that `user` is a message store keyword.

Examples

The following example backs up the entire message store to `/dev/rmt/0`:

```
imsbackup -f /dev/rmt/0 /
```

The following backs up the mailboxes of user ID `joe` to `/dev/rmt/0`:

```
imsbackup -f /dev/rmt/0 /primary/user/joe
```

The following example backs up all the mailboxes of all the users defined in the backup group groupA to backupfile:

```
imsbackup -f- /primary/groupA > backupfile
```

imsconnutil

Monitors user access of the message store. imsconnutil can provide the following information:

- Who is currently logged in on IMAP or Messenger Express (or any http web mail client).
- The last access time (log in or log out) for a specified user.
- For IMAP: lists the authentication method, the IP address from which the user is logged in, the IP address to which the user is connected, and the port on which they are logged to and from.

NOTE Do not kill this process while it is operating.

This command requires root access by the system user, and you may set the configuration variables local.imap.enableuserlist, local.http.enableuserlist, local.enablelastaccess to 1.

Location: *msg_svr_base*/sbin

Syntax

```
imsconnutil [-a|c] [-s service] [-u uid] [-f filename]
```

Options

The options for this command are:

Option	Description
-c|-a	One of -c or -a must be used.

Option	Description
`-a`	Last IMAP, POP, or http web mail client access (log in or log out) of user(s). `-s` does not affect the output of `-a`.
`-c`	List IMAP or Messenger Express users currently connected.
`-s` *service*	Can specify either `imap` or `http` as service to monitor. Only applies to `-c` option. POP is not available because POP users do not typically stay logged on.
`-u` *uid*	Specify a UID to monitor. If -u and -f are not listed, then all users are monitored.
`-f` *filename*	File containing UIDs to monitor. Each UID must be on its own line.
`--v`	Returns version of this tool.
`--h`	Returns usage information.

Examples

The following examples show `imsconnutil` and some various flags.

Lists every user ID currently logged into IMAP and http.

```
# imsconnutil -c
```

Lists last IMAP, POP, or Messenger Express access (log in or log out) of every user ID.

```
# imsconnutil -a
```

Lists access history (last log off or log on) of all user IDs. Lists current user IDs logged into IMAP and http.

```
# imsconnutil -a -c
```

Lists IMAP users currently logged on the message store.

```
# imsconnutil -c -s imap
```

Reveals whether user ID George is logged onto IMAP or not.

```
# imsconnutil -c -s imap -u George
```

Reveals whether user ID George is currently logged onto IMAP or Messenger Express, and lists the last time George was logged o or off.

```
# imsconnutil -c -a -u George
```

imsexport

The `imsexport` utility exports Sun Java System Messaging Server folders into UNIX `/var/mail` format folders.

The `imsexport` utility extracts the messages in a message store folder or mailbox and writes the messages to a UNIX file under the directory specified by the administrator. The file name is the same name as the IMAP folder name. For message store folders that contain both messages and sub-folders, `imsexport` creates a directory with the folder name and a file with the folder name plus a `.msg` extension. The *folder*`.msg` file contains the messages in the folder. The *folder* directory contains the sub-folders.

Location: *msg_svr_base*`/bin/msg/store/bin`

Syntax

```
imsexport -d dir -u user [-c y|n] [-e encoding] [-g] [-s mailbox] [-v mode]
```

Options

The options for this command are:

Option	Description
`-c y|n`	Provides an answer to the question: “Do you want to continue?” Specify y for yes, and specify n for no.
`-d` *dir*	Specifies the destination directory name where the folders will be created and written. This is a required option.

Option	Description
-e *encoding*	Specify an encoding option.
-g	Specifies debugging mode.
-s *mailbox*	Specifies the source folder to export.
-u *user*	Specifies the message store id for a user. Note that this is not necessarily the login id of the user. The message store id is either *userid* (for default domain users) or *userid@domain* (for other users). This is a required option.
-v *mode*	Specifies verbose mode. The values for *mode* are 0, 1, and 2. 0 specifies no output. 1 specifies mailbox level output. 2 (default) specifies message level output.

Example

In the following example, `imsexport` extracts all email for user `smith1`. `smith1` is a valid user account in the Sun Java System Messaging Server message store. User `smith1` has three folders on the store: `INBOX` (the normal default user folder), `private`, and `private/mom`. The destination directory will be `/tmp/joes_mail`.

```
% imsexport -u smith1 -d /tmp/joes_mail/
```

`imexport` then transfers each message store folder into a `/var/mail` conforming file. Thus you will get the following files:

- `/tmp/joes_mail/INBOX`
- `/tmp/joes_mail/private`
- `/tmp/joes_mail/private.msg`
- `/tmp/joes_mail/private/mom`

imsimport

The `imsimport` utility migrates UNIX `/var/mail` format folders into a Sun Java System Messaging Server message store.

The `imsimport` utility extracts the messages stored in `/var/mail` mailboxes and appends them to the corresponding users' mailbox in the Sun Java System Messaging Server message store. Files in the directory that are not in the standard UNIX mailbox format are skipped. If the corresponding users do not exist in the message store, `imsimport` creates them. When the user quota is exceeded, `imsimport` bypasses the message store quota enforcement, so the user does not receive an "over quota" message.

The `imsimport` utility can be run while Messaging Server is running. If mail delivery is enabled for the mailbox you are importing, old mail can get mixed with new mail, so you might want to hold the delivery of this user during the migration. Mailbox access should not be a problem.

NOTE `imsimport` does not use the IMAP server. However, the `stored` utility must be running to maintain message store integrity. The LDAP server should be running if `imsimport` is expected to create new users.

Location: *msg_svr_base*`/sbin/`

Syntax

```
imsimport -u user -s file [-c y|n] [-d mailbox] [-e encoding] [-g] [-n] [-v mode]
```

Options

The options for this command are:

Option	Description
`-c y|n`	Provides an answer to the question: "Do you want to continue?" Specify y for yes, and specify n for no.
`-d` *mailbox*	Specifies the destination mailbox where the messages will be stored.
`-e` *encoding*	Specify an encoding option.
`-g`	Specifies debugging mode.

Option	Description
-n	Creates a new mailbox with a *.date* extension if the mailbox exists. The *.date* extension is in the following form: *.mmddyy.HHMMSS* The month is specified by *mm*. The day is specified by *dd*. The year is specified by *yy*. For example, 052097 specifies May 20 in the year 1997. The time of day is specified by *HHMMSS*. For example 110000 specifies 11:00am.
-s *file*	Specifies the UNIX folder's file name where the messages are to be imported. The *file* parameter must be a full path name. This is a required option.
-u *user*	Specifies the message store id for a user. Note that this is not necessarily the login id of the user. This is a required option.
-v *mode*	Specifies verbose mode. The values for *mode* are 0, 1, and 2. 0 specifies no output. 1 specifies mailbox level output. 2 (default) specifies message level output.

Examples

`imsimport` migrates the specified `/var/mail/folder` for the specified user to the Sun Java System Messaging Server message store. If the destination folder is not specified, `imsimport` calls the destination folder by the same name as the source folder. In the following example, the command migrates the default `/var/mail` `INBOX` for the user `smith`, to the `INBOX`.

```
imsimport -u smith -s /var/mail/smith -d INBOX
```

Similarly, if you are trying to move a folder called test from `/home/smith/folders/` to the Sun Java System Messaging Server message store, use the following command:

```
imsimport -u smith -s /home/smith/folders/test -d test
```

If a destination folder called `test` already exists in the Sun Java System Messaging Server message store, `imsimport` appends the messages to the existing folder in the mailbox.

imsrestore

The `imsrestore` utility restores messages from the backup device into the message store.

Location: *msg_svr_base*`/sbin`

Syntax

```
imsrestore -f device|- [-a userid] [-b blocking_factor] [-c y | n] [-e encoding [-h]
[-i] [-m file] [-n] [-r file] [-t] [-u file] [-v 0|1|2] [path]
```

Options

The options for this command are:

Option	Description
`-b` *blocking_factor*	Indicates the blocking factor. Everything read on the device is performed by blocks of the size 512 x *blocking_factor*. The default is 20. Note: this number needs to be the same blocking factor that was used for the backup.
`-c y | n`	Automatically answers yes or no to the question “Do you want to continue?”
`-e` *encoding*	Mailbox name encoding (example: IMAP_MODIFIED-UTF-7)
`-f` *device*`|-`	When `-f-` is specified, backup data from `stdin` is read. Otherwise, the backup data is read from the specified device or filename.
`-h`	Dumps the header.
`-g`	Debug mode
`-i`	Ignores existing messages. Does not check for existing messages before restore. Note that if you specify the `-i` option, you may have duplicate messages after the restore, since the `-i` option supersedes your ability to check for duplicates.

Option	Description

`-m` *file*	This mapping file is used when renaming user IDs. The format in the mapping file is *oldname*=*newname* with one set of names per line. For example:
	`a=x`
	`b=y`
	`c=z`
	where `a, b,` and `c` are old names and `x, y,` and `z` are new names.
	This option is only used to rename user IDs from an older version of Messaging Server to a newer version of Messaging Server. Use the `-u` option for restoring users from SIMS to Messaging Server.

`-n`	Creates a new mailbox with a `.date` extension (if the mailbox exists). By default, messages are appended to the existing mailbox.
`-r` *file*	Reference file name (will restore all links in *file*).
`-t`	Prints a table of contents, but restore is not performed.

`-u` *file*	Specifies a backup object file. This file contains the object names (entire message store, user, group, mailbox, and so on) to restore. See *name* for a list of backup objects.
	For restoring SIMS data into a Sun Java System Message Store, you can specify or rename users with `-u` *file*. Users should have one name per line. If you rename users, the format of *file* is `oldname=newname` with one set of names per line. For example:
	`joe`
	`bonnie`
	`jackie=jackie1`
	where `joe` and `bonnie` are restored, and `jackie` is restored and renamed to `jackie1`. Note that full object pathnames are not needed for user IDs.

`-v [0|1|2]`	Executes the command in verbose mode. `0` = no output `1`= output at mailbox level (default) `2`= output at message level `3`=print meta data (for use with `-t` only) `4`=print object level meta data (for use with `-t` only)
name	Can be 1) logical pathname of the backup object, 2) user ID, 3) *mailbox*. See “imsbackup” on page 43 for description.

Examples

The following example restores the messages from the file `backupfile`:

```
imsrestore -f backupfile
```

The following example restores the messages for `joe` from the file `backupfile`:

```
imsrestore -f backupfile /primary/user/joe
```

The following example lists the content of the file `backupfile`:

```
imsrestore -f backupfile -t
```

The following example renames users in the file `mapfile`:

```
imsrestore -m mapfile -f backupfile
```

where the `mapfile` format is `oldname=newname`:

```
userA=user1
userB=user2
userC=user3
```

imscripter

The `imscripter` utility connects to an IMAP server and executes a command or a sequence of commands.

May be run remotely.

Location: *msg_svr_base*`/sbin/`

Syntax

```
imscripter [-h] [-f script | [-c command] -f datafile]] [-c command]
  [-s serverid | -p port | -u userid | -x passwd | -v verbosity]
```

Options

The options for this utility are:

Option	Description
`-c` *command*	Executes command, which can be one of the following: `create` *mailbox* `delete` *mailbox* `rename` *oldmailbox newmailbox* [*partition*] `getacl` *mailbox* `setacl` *mailbox userid rights* `deleteacl` *mailbox userid* If one or more of the above variables are included, the option executes the given command with that input. For example, `create lincoln` creates a mailbox for the user `lincoln`. If the `-f` *file* option is used, the option executes the command on each variable listed in the file.
`-f` *file*	The *file* may contain one or more commands, or a list of mailboxes on which commands are to be executed.
`-h`	Displays help for this command.
`-p` *port*	Connects to the given port. The default is 143.
`-s` *server*	Connects to the given server. The default is `localhost`. The server can be either a host name or an IP address.
`-u` *userid*	Connects as *userid*.

Option	Description
`-v` *verbosity*	String containing options for printing various information. The options are as follows: E—Show errors I—Show informational messages P—Show prompts C—Show input commands c—Show protocol commands B—Show BAD or NO untagged responses O—Show other untagged responses b—Show BAD or NO completion results o—Show OK completion results A—Show all of the above The letters designating options can be entered in any order. The default is `EPBibo`.
`-x` *passwd*	Uses this password.

mboxutil

The `mboxutil` command lists, creates, deletes, renames, or moves mailboxes (folders). `mboxutil` can also be used to report quota information.

You must specify mailbox names in the following format:

`user/`*`userid`*`/`*`mailbox`*

userid is the user that owns the mailbox and *mailbox* is the name of the mailbox.

Requirements: Must be run locally on the Messaging Server; the `stored` utility must also be running.

Location: *msg_svr_base*`/bin/msg/admin/bin`

Syntax

```
mboxutil [-c mailbox] [-d mailbox] [-f file]
  [-r oldname newname [partition]] [-l] [-o] [-p pattern] [-w file] [-x]
```

Options

The options for this command are:

Option	Description
`-c` *mailbox*	Creates the specified mailbox. A mailbox must exist before creating a secondary mailbox.
`-d` *mailbox*	Deletes the specified mailbox. There are some limitations with using the `-d` option to remove a user. Using the `-d` option to remove an active user (by removing the mailbox `user/`*userid*`/INBOX`) could result in a partially deleted mailbox. This occurs when either the user is connecting to the server or when the server is delivering mail to the user's mailbox. The recommended method to delete a user is to mark the user status as deleted in LDAP (using the `imadmin user delete` utility), and then use the `imadmin user purge` utility to purge the users that have been marked as deleted for a period longer than the specified number of days.

`-f` *file*	Creates, deletes the specified data file. The `-f` option can be used with the `-c`, `-d` or `-k` options.
	The data file contains a list of mailboxes on which the mboxutil command is executed. The following is an example of entries in a data file:
	`user/daphne/INBOX`
	`user/daphne/projx`
	`user/daphne/mm`

`-l`	Lists all of the mailboxes on a server. If you create multibyte folders for different language locales, you should edit: *msg_svr_base*`/bin/msg/bundles/encbylang.properties` to associate the appropriate character set with the LANG environment variable.
`-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, `mboxutil` writes the following command to the standard output: `mboxutil-d user/`*userid*`/INBOX` unless `-w` is specified

Option	Description
-p *pattern*	When used in conjunction with the -l option, lists only those mailboxes with names that match pattern. You can use IMAP wildcards. This option expects a pattern in IMAP M-UTF-7 format. This is not the recommended way to search for non ascii mailboxes. To search for non ascii mailboxes, use the -P option.
-P *regexp*	Lists only those mailboxes with names that match the specified POSIX regular expression. This option expects the *regexp* in the local language
-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.
-w *file*	Write command to file.
-x	When used in conjunction with the -l option, displays the path and access control for a mailbox.

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 Desdemona's mail folder from `memos` to `memos-april`:

```
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 the locked mail folder.

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:

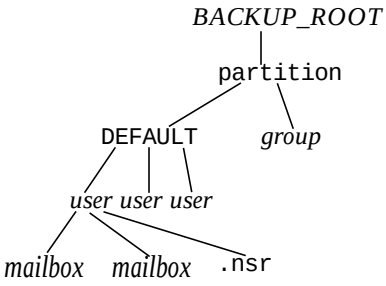
```
mboxutil -r user/dimitria/personal user/dimitria/personal \
partition
```

mkbackupdir

The `mkbackupdir` utility creates and synchronizes the backup directory with the information in the message store. It is used in conjunction with Solstice Backup (Legato Networker). The backup directory is an image of the message store. It does not contain the actual data. `mkbackupdir` scans the message store's user directory, compares it with the backup directory, and updates the backup directory with the new user names and mailbox names under the message store's user directory.

The backup directory is created to contain the information necessary for Networker to backup the message store at different levels (server, group, user, and mailbox). Figure 1-1 displays the structure.

Figure 1-1 Backup directory hierarchy



Location: *msg_svr_base*`/bin/msg/store/bin`

The variables in the backup directory contents are:

Variable	Description
BACKUP_ROOT	Message store administrator root directory.
partition	Store partition.

Variable	Description
group	System administrator-defined directories containing user directories. Breaking your message store into groups of user directories allows you to do concurrent backups of groups of user mailboxes. To create groups automatically, specify your groups in the *msg_svr_base*/`config/backup-groups.conf` file. The format for specifying groups is: *groupname*= *pattern* *groupname* is the name of the directory under which the user and mailbox directories will be stored, and *pattern* is a folder name with IMAP wildcard characters specifying user directory names that will go under the *groupname* directory.
user	Name of the message store user.
folder	Name of the user mailbox.
mailbox	Name of the user mailbox.

The `mkbackupdir` utility creates:

- A default *group* directory (`ALL`) or the group directories defined in the `backup-groups.conf` configuration file. The following is a sample `backup-groups.conf` file:

```
groupA=a* (regexp)
groupB=b*
groupC=c*
.
.
.
```

- A *user* directory under the backup directory for each new user in the message store.
- A 0 length mailbox file for each mailbox.
- A `.nsr` file for each subdirectory that contains user mailboxes.

The `.nsr` file is the NSR configuration file that informs the Networker to invoke `imsasm`. `imsasm` then creates and interprets the data stream.

Each user mailbox contains files of zero length. This includes the `INBOX`, which is located under the *user* directory.

NOTE Make sure the backup directory is writable by the message store owner (mailsrv).

Syntax

```
mkbackupdir [-a name_of_asm] [-i | -f] [-g] [-t number_of_threads] [-v] -p directory
```

Options

The options for this command are:

Option	Description
`-a` *name_of_asm*	Creates `.nsr` files using the specified asm name. This can be used for when you have multiple instances of Messaging Server as in symmetric HA environments.
`-f`	Backs up the folders only. By default, all mailboxes are backed up.
`-g`	Executes the command in debug mode.
`-i`	Backs up the inbox only. By default, all mailboxes are backed up.
`-p` *directory*	Specifies the directory for the backup image. This is a required option. Note: The Networker has a limitation of 64 characters for *saveset* name. If your default backup directory pathname is too long, you should use this option to specify another pathname.
`-t` *number_of_threads*	Specifies the number of threads that can be used to create the backup directory. The default is one thread for each partition, which is usually adequate. If you have many partitions, and you do not want `mkbackupdir` to consume all your resources, you can lower this number.
`-u`	User level backup. Instead of backing up each folder as a file, create a backup file per user.
`-v`	Executes the command in verbose mode.

Examples

To create the mybackupdir directory, enter the following:

```
mkbackupdir -p /mybackupdir
```

MoveUser

The MoveUser utility moves a user's account from one messaging server to another. When user accounts are moved from one messaging server to another, it is also necessary to move the user's mailboxes and the messages they contain from one server to the other. In addition to moving mailboxes from one server to another, MoveUser updates entries in the directory server to reflect the user's new mailhost name and message store path.

May be run remotely.

Location: *msg_svr_base*/sbin/

NOTE If you expect the moveuser utility to alter the LDAP attributes, then you must run the following command to set the authentication cache timeout value to 0:

```
configutil -o service.authcachettl -v 0
```

Syntax

```
MoveUser -s srcmailhost[:port] -x proxyuser -p password -d destmailhost[:port]
  [-u uid | -u uid -U newuid| -l ldapURL -D binDN -w password
  [-r DCroot -t defaultDomain]] [-a destproxyuser]
```

Options

The options for this command are:

Option	Description
-a *destproxyuser*	ProxyAuth user for destination messaging server.
-A	Do not add an alternate email address to the LDAP entry.

Option	Description
`-d` *destmailhost*	Destination messaging server. By default, `MoveUser` assumes IMAP port `143`. To specify a different port, add a colon and the port number after *destmailhost*. For example, to specify port `150` for `myhost`, you would enter: `-d myhost:150`
`-D` *binddn*	Binding *dn* to the given *ldapURL*.
`-F`	Delete messages in source messaging server after successful move of mailbox. (If not specified, messages will be left in source messaging server.)
`-h`	Display help for this command.
`-l` *ldapURL*	URL to establish a connection with the Directory Server: `ldap://`*hostname*`:`*port*`/`*base_dn*`?`*attributes*`?`*scope*`?`*filter* For more information about specifying an LDAP URL, see your Directory Server documentation. Cannot be used with the `-u` option.
`-L`	Add a license for Messaging Server if not already set.
`-m` *destmaildrop*	Message store path for destination messaging server. (If not specified, the default is used.)
`-n` *msgcount*	Number of messages to be moved at once.
`-o` *srcmaildrop*	Message store path for source messaging server. (If not specified, the default is used.)
`-p` *srcproxypasswd*	ProxyAuth password for source messaging server.
`-r` *DCroot*	DC root used with the `-l` option to move users under a hosted domain.
`-s` *srcmailhost*	Source messaging server. By default, `MoveUser` assumes IMAP port `143`. To specify a different port, add a colon and the port number after *srcmailhost*. For example, to specify port `150` for `myhost`, you would enter: `-s myhost:150`.
`-S`	Do not set new message store path for each user.
`-t` *defaultDomain*	Default domain used with the `-l` option to move users under a hosted domain.
`-u` *uid*	User ID for the user mailbox that is to be moved. Cannot be used with `-l` option.

Option	Description
-U *newuid*	New (renamed) user ID that the mailbox is to be moved to. Must be used with -u *uid*, where -u *uid*, identifies the old user name that is to be discontinued. Both the old and the new user ID must currently exist on both the source and the destination mailhost. After migration you are free to manually remove the original user ID from LDAP if you wish to do so.
-v *destproxypwd*	ProxyAuth password for destination messaging server.
-w *bindpasswd*	Binding password for the *binddn* given in the -D option.
-x *srcproxyuser*	ProxyAuth user for source messaging server.

Examples

To move all users from host1 to host2, based on account information in the Directory Server siroe.com:

```
MoveUser -l \
"ldap://siroe.com:389/o=siroe.com???(mailhost=host1.domain.com)" \
-D "cn=Directory Manager" -w password -s host1 -x admin \
-p password -d host2 -a admin -v password
```

To move one user from host1 which uses port 150 to host2, based on account information in the Directory Server siroe.com:

```
MoveUser -l \
"ldap://siroe.com:389/o=siroe.com???(uid=userid)" \
-D "cn=Directory Manager" -w password -s host1:150 -x admin \
-p password -d host2 -a admin -v password
```

To move a group of users whose uid starts with letter ‘s’ from host1 to host2, based on account information in the Directory Server server1.siroe.com:

```
MoveUser -l \
"ldap://server1.siroe.com:389/o=siroe.com???(uid=s*)" \
-D "cn=Directory Manager" -w password -s host1 -x admin \
-p password -d host2 -a admin -v password
```

To move a user's mailboxes from `host1` to `host2` when the user ID of `admin` is specified in the command line:

```
MoveUser -u uid -s host1 -x admin -p password -d host2 -a admin \
-v password
```

To move a user named `aldonza` from `host1` to a new user ID named `dulcinea` on `host2`:

```
MoveUser -u aldonza -U dulcinea -s host1 -x admin -p password \
-d host2 -a admin -v password
```

msuserpurge

When user and domain mailboxes marked for deletion, the `msuserpurge` command purges those user and domain mailboxes from the message store. Specifically, this command scans the following domain and user status attributes in LDAP for a value of `deleted`: `inetDomainStatus`, `mailDomainStatus`, `inetUserStatus`, `mailUserStatus`. This command can be run at the command line, or can be scheduled for execution with the `configutil` parameter `local.sched.userpurge`.

Requirements: If run manually, it must be manually run locally on the messaging server. Make sure that the environment variable `CONFIGROOT` is set to *msg_svr_base*`/config`.

Location: *msg_svr_base*`/lib`

Syntax

```
msuserpurge [-d domain_name] [-g grace_period]
```

Options

The options for this command are:

Option	Description
`-d` *domain_name*	Specifies domain to check for `deleted` attribute, and, if set, purges the mailboxes in that domain. If `-d` is not specified, then all domains on this mail host are checked for the `deleted` attribute and all mailboxes in the deleted domains are purged. If the domain spans more than one mail host, then you need to run this command on each host. (This command uses the `mailhost` attribute to determine where to purge.)
`-g` *grace_period*	Specifies the number of days that a domain or user must be marked as deleted before this command is run.

Examples

```
msuserpurge -d siroe.com
```

readership

An owner of an 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 or have access rights to shared folders.

Requirements: Must be run locally on the messaging server; the `stored` utility must also be running.

Location: *msg_svr_base*`/sbin/`

Syntax

```
readership [-d days] [-p months] [-l] [-s folder identifier right]
```

Options

The options for this command are:

Option	Description
`-d` *days*	Counts as a reader any identity that has selected the shared IMAP folder within the indicated number of days. The default is 30.
`-p` *months*	Does not count users who have not selected the shared IMAP folder within the indicated number of months. The default is infinity and removes the seen flag data for those users. This option also removes the “seen” flag data for those users from the store.
`-l`	List the data in `lright.db`.
`-s` *folder identifier right*	Setacl for folder.

reconstruct

The `reconstruct` utility rebuilds one or more mailboxes, or the master mailbox file (the mailboxes database), and repairs any inconsistencies. You can use this utility to recover from almost any form of data corruption in the message store.

A mailbox consists of files under the user partition directory. The mailboxes database is the mboxlist database.

Requirements: Must be run locally on the messaging server; the `stored` utility must also be running.

Location: *msg_svr_base*`/sbin/`

NOTE Low-level database repair, such as completing transactions and rolling back incomplete transactions is performed with `stored -d`.

Syntax

```
reconstruct [-n | -f] [-l] [-p partition] -r [mailbox [mailbox...]]

reconstruct [-n | -f] [-p partition] mailbox [mailbox...]

reconstruct [-p partition] -m

reconstruct -q
```

Options

The options for this command are:

Option	Description
`-f`	Forces `reconstruct` to perform a fix on the mailbox or mailboxes.
`-l`	Reconstruct `lright.db`.
`-m`	Repairs and performs a consistency check of the mailboxes database. This option examines every mailbox it finds in the spool area, adding or removing entries from the mailboxes database as appropriate. The utility prints a message to the standard output file whenever it adds or removes an entry from the database.

`-n`	Checks the message store only, without performing a fix on the mailbox or mailboxes. The `-n` option cannot be used by itself, unless a mailbox name is provided. When a mailbox name is not provided, the `-n` option must be used with the `-r` option; the `-r` option may be combined with the `-p` option. For example, any of the following commands are valid:
	`reconstruct -n user/dulcinea/INBOX`
	`reconstruct -n -r`
	`reconstruct -n -r -p primary`
	`reconstruct -n -r user/dulcinea/`

`-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.

The *mailbox* argument indicates the mailbox to be repaired. You can specify one or more mailboxes. Mailboxes are specified with names in the format `user/`*userid*`/`*sub-mailbox*, where *userid* is the user that owns the mailbox. For example, the inbox of the user `dulcinea` is entered as: `user/dulcinea/INBOX`.

Examples

The following command performs a reconstruct on a specific mailbox:

```
reconstruct user/dulcinea/INBOX
```

The following checks the specified mailbox, without performing a reconstruct:

```
reconstruct -n user/dulcinea/INBOX
```

The following command checks all mailboxes in the message store:

```
reconstruct -n -r
```

refresh

The refresh utility refreshes the configuration of the specified messaging server processes (SMTP, IMAP, POP, STORE, HTTP, ENS, SCHED). It is used when an option for one of the services has been modified and you wish this option to take effect.

Location: *msg_svr_base*/sbin

Syntax

```
refresh [dispatcher | job_controller | smtp | imap | pop | store | http | ens
| sched]
```

Examples

The following command refreshes the scheduler utility:

```
refresh sched
```

If refresh does not cause the change to take effect, then stop and restart the service.

start-msg

The `start-msg` utility starts all of the messaging server processes (`smtp`, `imap`, `pop`, `store`, `http`, `ens`, `sched`), or optionally, one specified service. The services started by start-msg 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`.

Location: *msg_svr_base*`/sbin`

Syntax

```
start-msg [dispatcher | job_controller | smtp | imap | pop | store | http |
ens | sched] [snmp] [sms] [mmp]
```

Examples

The following command starts all the Messaging Server processes:

```
start-msg [-a]
```

The following command starts the `imap` process:

```
start-msg imap
```

`-a : ha mode`

stop-msg

The `stop-msg` utility stops all Messaging Server processes (`smtp`, `imap`, `pop`, `store`, `http`, `ens`, `sched`), or optionally, one specified service. To use `stop-msg` *component*, the component must be enabled. The `stop-msg` command without arguments shuts down everything started by `start-msg`, including disabled components.

Location: *msg_svr_base*`/sbin`

Syntax

```
stop-msg [dispatcher | job_controller | smtp | imap | pop | store | http |
ens | sched] [snmp] [sms] [mmp]
```

Examples

The following command stops all Messaging Server processes:

```
stop-msg
```

The following command stops the `http` service:

```
stop-msg http
```

stored

The `stored` utility performs the following functions:

- Background and daily messaging tasks
- Deadlock detection and rollback of deadlocked database transactions

Requirements: Must be run locally on the Messaging Server.

Location: *msg_svr_base*`/lib/`

Syntax

To run `stored` as a daemon process:

```
stored [-r] [-R] [-t] [-v]
```

Options

The options for this command are:

Option	Description
`-r`	Removes the database temporary files and synchronizes the database. This cleans up the database environment to prepare for an upgrade, downgrade, or migration.
`-R`	Removes the database temporary files without synchronizing the database. This is used if the `-r` option fails because of a corrupted database. This forces removal of temporary files.
`-t`	Checks the status of `stored`. The return code of this command indicates the status. To print the status, enter: `stored -t -v`
`-v`	Verbose output.

Chapter 2

Message Transfer Agent Command-line Utilities

The command-line utilities described in this chapter are used to perform various maintenance, testing, and management tasks for the Message Transfer Agent (MTA).

The MTA commands are also referred to as the `imsimta` commands. The `imsimta` script is located in the *msg_svr_base*`/` directory.

msg_svr_base represents the directory path in which you install the server.

The commands are listed in Table 2-1

Table 2-1 MTA Commands

Command	Description
imsimta cache	Performs operations on the queue cache.
imsimta chbuild	Compiles the MTA character set conversion tables.
imsimta cnbuild	Compiles the MTA configuration files.
imsimta counters	Performs operations on the channel counters.
imsimta crdb	Creates an MTA database.
imsimta find	Locates the precise filename of the specified version of an MTA log file
imsimta kill	Terminates the specified process.
imsimta process	Lists currently running MTA jobs.
imsimta program	Manipulates the MTA program delivery options.
imsimta purge	Purges MTA log files.
imsimta qclean	Holds or deletes message files containing specific substrings in their envelope From:address, Subject: line, or content.
imsimta qm	Manages MTA message queues.

Table 2-1 MTA Commands *(Continued)*

Command	Description
imsimta qtop	Displays the most frequently occurring envelope From: Subject:, or message content fields found in message files in the channel queues.
imsimta refresh	Combines the functionality of the imsimta cnbuild and imsimta restart utilities.
imsimta reload	Allows changes to certain configuration files to take effect without restarting the server.
imsimta renamedb	Renames a MTA database.
imsimta restart	Restarts detached MTA processes.
imsimta return	Returns (bounces) a mail message to its originator.
imsimta run	Processes messages in a specified channel.
imsimta start	Starts the MTA Job Controller and Dispatcher.
imsimta stop	Shuts down the MTA Job Controller and the MTA Dispatcher.
imsimta submit	Processes messages in a specified channel.
imsimta test	Performs tests on mapping tables, wildcard patterns, address rewriting, and URLs.
imsimta version	Prints the MTA version number.
imsimta view	Displays log files.

Command Descriptions

You need to be logged in as root (UNIX) or administrator (Windows NT) to run the MTA commands. Unless mentioned otherwise, all MTA commands should be run as mailsrv (the mail server user that is created at installation).

imsimta cache

The MTA maintains an in-memory cache of all the messages currently stored in its queues. This cache is called the queue cache. The purpose of the queue cache is to make dequeue operations perform more efficiently by relieving master programs from having to open every message file to find out which message to dequeue and in which order.

Syntax

```
imsimta cache -sync | -view [channel]
```

Options

The options for this command are:

Option	Description
`-sync`	Updates the active queue cache by updating it to reflect all non-held message files currently present in the `/`*msg_svr_base*`/imta/queue/` subdirectories. Note that the `-sync` option does not remove entries from the queue cache. The queue cache entries not corresponding to an actual queued message are silently discarded by channel master programs.
`-view` [*channel*]	Shows the current non-held entries in the MTA queue cache for a channel. channel is the name of the channel for which to show entries. This is a potentially expensive command if you have a large backlog of messages. Instead of running this command frequently, consider using the imsimta qm messages command.
`-walk [-debug=xxx]`	Used to diagnose potential problems with the job controller. Each time this command is run, the internal state of the message queues and job scheduling information is written to the `job_controller.log` file. In addition, the job controller debug mask is set to the value given. You should not run this command unless you are instructed to do so by support.

Examples

To synchronize the queue cache:

```
imsimta cache -sync
```

To view entries in the queue cache for the `tcp_local` channel, execute the command:

```
imsimta cache -view tcp_local
```

imsimta chbuild

The `imsimta chbuild` command compiles the character set conversion tables and loads the resulting image file into shared memory. The MTA ships with complete character set tables so you would not normally need to run this command. You would use `imsimta chbuild` only if you added or modified any character sets.

Syntax

```
imsimta chbuild [-image_file=file_spec | -noimage_file]
  [-maximum | -nomaximum]
  [-option_file=[option_file] | -nooption_file] [-remove]
  [-sizes |-nosizes]  [-statistics | -nostatistics]
```

Options

The options for this command are:

Option	Description
`-image_file=`*file_spec* | `-noimage_file`	By default, `imsimta chbuild` creates as output the image file named by the `IMTA_CHARSET_DATA` option of the MTA tailor file, *msg_svr_base*`/config/imta_tailor`. With the `-image_file` option, an alternate file name may be specified. When the `-noimage_file` option is specified, `imsimta chbuild` does not produce an output image file. The `-noimage_file` option is used in conjunction with the `-option_file` option to produce as output an option file that specifies table sizes adequate to hold the tables required by the processed input files.
`-maximum` | `-nomaximum`	The file *msg_svr_base*`/config/maximum_charset.dat` is read in addition to the file named by the `IMTA_CHARSET_OPTION_FILE` option of the MTA tailor file, *msg_svr_base*`/config/imta_tailor`, when `-maximum` is specified. This file specifies near `-maximum` table sizes but does not change any other settings. Use this option only if the current table sizes are inadequate. The `-noimage` and `-option_file` options should always be used in conjunction with this option—it makes no sense to output the enormous configuration that is produced by `-maximum`, but it does make sense to use `-maximum` to get past size restrictions in order to build a properly sized option file for use in building a manageable configuration with a subsequent `imsimta chbuild` invocation.

Option	Description
-option_file=[*option_file*] | -nooption_file	imsimta chbuild can produce an option file that contains the correct table sizes to hold the conversion tables that were just processed (plus a little room for growth). The -option_file option causes this file to be output. By default, this file is the file named by the IMTA_CHARSET_OPTION_FILE option of the MTA tailor file, *msg_svr_base*/config/imta_tailor. The value of the -option_file option may be used to specify an alternate file name. If the -nooption_file option is given, then no option file is output. imsimta chbuild always reads any option file (for example, the file named by the IMTA_OPTION_FILE option of the MTA tailor file) that is already present; use of this option does not alter this behavior. However, use of the -maximum option causes imsimta chbuild to read options from maximum_charset.dat in addition to IMTA_CHARSET_OPTION_FILE. This file specifies near-maximum table sizes. Use this option only if the current table sizes are inadequate, and only use it to create a new option file. The -noimage_file option should always be specified with -maximum, since a maximum-size image would be enormous and inefficient.
-remove	Removes any existing compiled character set conversion table. This is the file named by the IMTA_CHARSET_DATA option of the MTA tailor file, *msg_svr_base*/config/imta_tailor.
-sizes | -nosizes	The -sizes option instructs imsimta chbuild to output or suppress information on the sizes of the uncompiled conversion tables. The -nosizes option is the default.
-statistics | -nostatistics	The -statistics option instructs imsimta chbuild to output or suppress information on the compiled conversion tables. This information gives a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the -option_file option is needed. The -nostatistics option is the default.

Example

The standard command you use to compile character set conversion tables is:

```
imsimta chbuild
```

imsimta cnbuild

The `imsimta cnbuild` command compiles the textual configuration, option, mapping, conversion, circuit check and alias files, and loads the resulting image file into shared memory. The resulting image is saved to a file usually named `imta/lib/config_data` by the `IMTA_CONFIG_DATA` option of the MTA tailor file, *msg_svr_base*`/config/imta_tailor`.

Whenever a component of the MTA (for example, a channel program) must read a compiled configuration component, it first checks to see whether the file named by the MTA tailor file option `IMTA_CONFIG_DATA` is loaded into shared memory; if this compiled image exists but is not loaded, the MTA loads it into shared memory. If the MTA finds (or not finding, is able to load) a compiled image in shared memory, the running program uses that image.

The reason for compiling configuration information is simple: performance. The only penalty paid for compilation is the need to recompile and reload the image any time the underlying configuration files are edited. Also, be sure to restart any programs or channels that load the configuration data only once when they start up-for example, the MTA multithreaded SMTP server.

It is necessary to recompile the configuration every time changes are made to any of the following files:

- MTA configuration file (or any files referenced by it)
- MTA system alias file
- MTA mapping file
- MTA option file
- MTA conversion file
- MTA security configuration file
- MTA circuit check configuration file
- MTA system wide filter file

Specifically, these are the files pointed at by the MTA tailor file options `IMTA_CONFIG_FILE`, `IMTA_ALIAS_FILE`, `IMTA_MAPPING_FILE`, `IMTA_OPTION_FILE`, and `IMTA_CONVERSION_FILE` respectively, which usually point to the following files:

- *msg_svr_base*`/config/imta.cnf`
- *msg_svr_base*`/config/aliases`
- *msg_svr_base*`/config/mappings`

- *msg_svr_base*/config/option.dat
- *msg_svr_base*/config/conversions

NOTE Until the configuration is rebuilt, changes to any of these files are not visible to the running MTA system.

Syntax

```
imsimta cnbuild [-image_file=file_spec | -noimage_file]
  [-maximum | -nomaximum]
  [-option_file=[option_file] | -nooption_file] [-remove]
  [-sizes | -nosizes] [-statistics | -nostatistics]
```

Options

The options for this command are:

Option	Description
-image_file=*file_spec* | -noimage_file	By default, imsimta cnbuild creates as output the image file named by the IMTA_CONFIG_DATA option of the MTA tailor file, *msg_svr_base*/config/imta_tailor. With the -image_file option, an alternate filename can be specified. When the -noimage_file option is specified, imsimta cnbuild does not produce an output image file. This option is used in conjunction with the -option_file option to produce as output an option file which specifies table sizes adequate to hold the configuration required by the processed input files. The default value is -image_file=IMTA_CONFIG_DATA.
-maximum | -nomaximum	*msg_svr_base*/config/maximum.dat is read in addition to the file named by the IMTA_OPTION_FILE option in the MTA tailor file, *msg_svr_base*/config/imta_tailor. This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this option if the current table sizes are inadequate. The -noimage and -option_file options should always be used in conjunction with this qualifier; it makes no sense to output the enormous configuration that is produced by -maximum, but it does make sense to use -maximum to get past size restrictions in order to build a properly-sized option file so that a proportionately-sized configuration can be built with a subsequent imsimta cnbuild invocation. The default is -nomaximum.

Option	Description
-option_file=[*option_file*] | -nooption_file	imsimta cnbuild can optionally produce an option file that contains correct table sizes to hold the configuration that was just compiled (plus a little room for growth). The -option_file option causes this file to be output. By default, this file is the file named by the IMTA_OPTION_FILE option in the MTA tailor file, *msg_svr_base*/config/imta_tailor. The value on the -option_file option may be used to specify an alternate file name. If the -nooption_file option is given, then no option file will be output. imsimta cnbuild always reads any option file that is already present via the IMTA_OPTION_FILE option of the MTA tailor file, *msg_svr_base*/config/imta_tailor; use of this option will not alter this behavior. However, use of the -maximum option causes imsimta cnbuild to read MTA options from the *msg_svr_base*/config/maximum.dat file in addition to reading the file named by IMTA_OPTION_FILE. This file specifies near maximum table sizes. Use this option only if the current table sizes are inadequate, and only to create a new option file. The -noimage_file option should always be specified when -maximum is specified since a maximum-size image would be enormous and wasteful. The default value is -option_file=IMTA_OPTION_FILE.
-remove	Remove any existing compiled configuration; for example, remove the file named by the IMTA_CONFIG_DATA option of the MTA tailor file, *msg_svr_base*/config/imta_tailor.
-sizes | -nosizes	The -sizes option instructs imsimta cnbuild to output information on the sizes of uncompiled MTA tables. The -nosizes option is the default.
-statistics | -nostatistics	The -statistics option instructs imsimta cnbuild to output information table usage. This information gives a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the -resize_tables option is needed. The -nostatistics option is the default.

Examples

To regenerate a compiled configuration enter the following command:

```
imsimta cnbuild
```

After compiling the configuration, restart any programs that may need to reload the new configuration. For example, the SMTP server should be restarted:

```
imsimta restart dispatcher
```

NOTE `imsimta cnbuild` is executed whenever the `imsimta refresh` command is invoked.

imsimta counters

The MTA accumulates message traffic counters for each of its active channels. These statistics, referred to as channel counters, are kept in shared memory. The `imsimta counters` command manipulates these counters.

Syntax

```
imsimta counters -clear

imsimta counters -create [-max_channels=value]

imsimta counters -delete

imsimta counters -show [-associations | -noassociations]
  [-channels | -nochannels] [-headers | -noheaders]
  [-output=file_spec]
```

Options

The options for this command are:

Option	Description
`-associations` | `-noassociations`	Specifies whether or not to show the in-memory cache of association counters. The `-associations` option is the default. This option is only used with the `-show` option.
`-channels` | `-nochannels`	Specifies whether or not to show the in-memory cache or channel counters. The `-channels` option is the default. This option is only used with the `-show` option.

Option	Description
`-clear`	The `-clear` command clears the in-memory channel counters.
`-create`	Creates the in-memory channel counters. Counters are not created by default. You must create the counters if you wish to use them. You must create them after restarting the MTA as well.
`-headers` | `-noheaders`	Controls whether or not a header line describing each column in the table of counters is output. The `-headers` option is the default. This option is only used with the `-show` option.
`-max_channels=`*value*	By default, the in-memory channel counters can hold information for `CHANNEL_TABLE_SIZE` channels. `CHANNEL_TABLE_SIZE` is the value specified by the MTA option file option of the same name. Use the `-max_channels=`*value* option to select a different size. This option is used only with the `-create` option.
`-delete`	Deletes the in-memory channel counters.
`-show`	Displays the in-memory channel counters.
`-headers` | `-noheaders`	Controls whether or not a header line describing each column in the table of counters is output. The `-headers` option is the default. This option is only used with the `-show` option.
`-output=`*file_spec*	Directs the output to the specified file. By default, the output appears on your display. This option is only used with the `-show` option.

Examples

To display the counters for all channels:

```
imsimta counters -show
```

imsimta crdb

The `imsimta crdb` command creates and updates MTA database files. `imsimta crdb` converts a plain text file into MTA database records; from them, it either creates a new database or adds the records to an existing database.

In general, each line of the input file must consist of a left side and a right side. The two sides are separated by one or more spaces or tabs. The left side is limited to 32 characters in a short database (the default variety) and 80 characters in a long database. The right side is limited to 80 characters in a short database and 256 in a long database. Spaces and tabs may not appear in the left side unless the `-quoted` option is specified. Comment lines may be included in input files. A comment line is a line that begins with an exclamation mark (!) in column 1.

Syntax

```
imsimta crdb input-file-spec output-database-spec [-append | -noappend]
  [-count | -nocount] [-duplicates | -noduplicates]
  [-long_records | -nolong_records] [-quoted | -noquoted]
  [-remove | -noremove] [-statistics | -nostatistics]
  [-strip_colons | -nostrip_colons]
```

Options

The options for this command are:

Option	Description
input-file-spec	A text file containing the entries to be placed into the database. Each line of the text file must correspond to a single entry. This attribute is mandatory.
output-database-spec	The initial name string of the files to which to write the database (unless `-dump` is specified). The `.db` extension is appended to the file name. This attribute is mandatory.
`-append` | `-noappend`	When the default, `-noappend`, option is in effect, a new database is created, overwriting any old database of that name. Use the `-append` option to instruct the MTA to instead add the new records to an existing database. The `-noappend` option is the default. In the event of a duplicate record, the newly appended record overwrites the old record when `-noduplicates` is specified.
`-count` | `-nocount`	Controls whether or not a count is output after each group of 100 input lines are processed. The `-count` option is the default.
`-duplicates` | `-noduplicates`	Controls whether or not duplicate records are allowed in the output files. Currently, duplicate records are of use only in the domain database (rewrite rules database) and databases associated with the directory channel. The `-noduplicates` option is the default.

Option	Description
-long_records | -nolong_records	Controls the size of the output records. By default, left sides are limited to 32 characters and right sides are limited to 80 characters. If -long_records is specified, the limits are changed to 80 and 256, respectively. The -nolong_records option is the default.
-quoted | -noquoted	Controls the handling of quotes. Normally imsimta crdb pays no attention to double quotes. If -quoted is specified, imsimta crdb matches up double quotes in the process of determining the break between the left and right hand sides of each input line. Spaces and tabs are then allowed in the left side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. The quotes are not removed unless the -remove option is also specified. The -noquoted option is the default.
-remove | -noremove	Controls the removal of quotes. If imsimta crdb is instructed to pay attention to quotes, the quotes are normally retained. If -remove is specified, imsimta crdb removes the outermost set of quotes from the left hand side of each input line. Spaces and tabs are then allowed in the left side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. -remove is ignored if -quoted is not in effect. The -noremove option is the default.
-statistics | -nostatistics	Controls whether or not some simple statistics are output by imsimta crdb, including the number of entries (lines) converted, the number of exceptions (usually duplicate records) detected, and the number of entries that could not be converted because they were too long to fit in the output database. -nostatistics suppresses output of this information. The -statistics option is the default.
-strip_colons | -nostrip_colons	Instructs imsimta crdb to strip a trailing colon from the right end of the left hand side of each line it reads from the input file. This is useful for turning alias file entries into an alias database. The -nostrip_colons is the default.

Example

The following commands create an alias database with “long” record entries. The creation is performed in a two-step process using a temporary database to minimize any window of time, such as during database generation, when the database would be locked and inaccessible to the MTA.

```
imsimta crdb -long_records aliases-tmp

imsimta renamedb aliases-tmp IMTA_ALIAS_DATABASE
```

imsimta crdb -dump

The `imsimta crdb -dump` command writes the entries in MTA databases to a flat ASCII file. In particular, this command may be used to write the contents of an old style database to a file from which a new style database may be built using the `imsimta crdb` command. The output begins with a comment line that displays a proper `imsimta crdb` command to use in order to return the ASCII output to a database.

NOTE Make sure you are logged in as `mailsrv` (the mail server user) before performing this command.

Syntax

```
imsimta crdb -dump input-database-spec [output-file-spec]
```

Parameters

The parameters for this command are:

Parameter	Description
input-database-spec	Database from which to read entries. By default, the MTA looks for a current format database of the given name; if this does not exist, the MTA will look for an old format database of the given name. The special keywords `IMTA_ALIAS_DATABASE`, `IMTA_REVERSE_DATABASE`, and `IMTA_GENERAL_DATABASE` are supported; the use of such a special keyword tells the MTA to dump the database specified by the corresponding MTA tailor file option.
output-file-spec	ASCII file to which the entries stored in the database are written. This file should be in a directory where you have write permissions. If an output file is not specified, the output is written to `stdout`.

Examples

The following command can be used to dump the contents of an alias database to a file, and then to recreate the alias database from that file

```
imsimta crdb -dump IMTA_ALIAS_DATABASE alias.txt
imsimta crdb alias.txt alias-tmp
imsimta renamedb alias-tmp IMTA_ALIAS_DATABASE
```

imsimta find

The `imsimta find` utility locates the precise filename of the specified version of an MTA log file. MTA log files have a *-uniqueid* appended to the filename to allow for the creation of multiple versions of the log file. On UNIX, the *-uniqueid* is appended to the very end of the filename (the end of the file extension), while on Windows NT, the *-uniqueid* is appended to the end of the name part of the filename, before the file extension. The `imsimta find` utility understands these unique ids and can find the particular filename corresponding to the requested version of the file.

Syntax

```
imsimta find file-pattern [-f=offset-from-first] [-l=offset-from-last]
```

Options

The options for this command are:

Option	Description
`-f=`*offset-from-first*	Finds the specified version of the file (starting from 0). For example, to find the earliest (oldest) version of the file, specify `-f=0`. By default, `imsimta find` finds the most recent version of the file.
`-l=`*offset-from-last*	Finds the last version of the specified file. For example, to find the most recent (newest) version of the file, specify `-l=0`. By default, `imsimta find` finds the most recent version of the file.
file-pattern	Specifies a filename pattern for which the log file to find.

Examples

The following command prints out the filename of the `tcp_local_slave.log-`*uniqueid* file most recently created:

```
imsimta find msg_svr_base/imsimta/log/tcp_local_slave.log
```

The following command displays the filename of the oldest tcp_bitnet_master.log-uniqueid file:

```
imsimta find \
msg_svr_base/imsimta/log/tcp_bitnet_master.log -f=0
```

imsimta kill

The `imsimta kill` utility immediately and indiscriminately terminates the specified process. This command is equivalent to the UNIX `kill -9` command. The process is terminated even if it is in the middle of transferring email. So use of the `imsimta shutdown` utility, which performs an orderly shutdown, is generally preferable.

Syntax

```
imsimta kill component
```

NOTE You must have the same process id as the process to be killed, or be `root`. This utility is not available on Windows NT.

component is the MTA component to be killed. Valid values are `job_controller` and `dispatcher`.

imsimta process

This command displays the current MTA processes. Additional processes may be present if messages are currently being processed, or if certain additional MTA components are in use.

Syntax

```
imsimta process
```

Example

The following command shows current MTA processes:

```
# imsimta process
```

```
imsimta process

USER     PID   S VSZ   RSS  STIME        TIME         COMMAND
mailsrv  15334 S 21368 9048 17:32:44     0:01
/export/ims/bin/msg/imta/bin/dispatcher
mailsrv  15337 S 21088 10968 17:32:45    0:01
/export/ims/bin/msg/imta/bin/tcp_smtp_server
mailsrv  15338 S 21080 11064 17:32:45    0:01
/export/ims/bin/msg/imta/bin/tcp_smtp_server
mailsrv  15349 S 21176 10224 17:33:02    0:02
/export/ims/bin/msg/imta/bin/job_controller
```

imsimta program

The `imsimta program` command is used to manipulate the program delivery options.

This command can be executed as `root` or `mailsrv`. `mailsrv` is the default user for Messaging Server, but could be whatever the specified user name for the Messaging Server is when Messaging Server is installed.

The program is passed the entire message, unparsed from `stdin`. This includes the From line (without the colon) as the first line, followed by the headers and the message body. This may include any MIME attachments that are part of the message.

Syntax

```
imsimta program -a -m method -p program [-g argument_list]
  [-e exec_permission]

imsimta program -d -m method

imsimta program -c -m method -p program | -g argument_list |
  -e exec_permission
```

Options

The options for this command are:

Option	Description
`-a`	Add a method to the set of program delivery methods. This option cannot be used with the `-d`, `-c`, `-l`, or `-u` options.
`-c`	Change the arguments to a program that has already been entered.
`-m` *method*	Name given by the administrator to a particular method. This will be the name by which the method will be advertised to users. Method names must not contain spaces, tabs, or equal signs (=). The method name cannot be none or local. The method name is restricted to U.S. ASCII. This option is required with the `-a`, `-d`, `-c`, and `-u` options.
`-p` *program*	Actual name of the executable for a particular method. The executable should exist in the programs directory (*msg_svr_base*`/data/site-programs`) for the add to be successful. It can be a symbolic link to an executable in some other directory. This option is required with the `-a` option.
`-g` *argument_list*	Argument list to be used while executing the program. If this option is not specified during an add, no arguments will be used. Each argument must be separated by a space and the entire argument list must be given within double quotes. If the %s tag is used in the argument list, it will be substituted with the user's username for programs executed by the users and with *username+programlabel* for programs executed by inetmail. *programlabel* is a unique string to identify that program. This option can be used with the `-a` and `-c` options.
`-e` *exec_permission*	*exec_permission* can be `user` or `postmaster`. If it is specified as user, the program is executed as the user. By default, execute permission for all programs are set to postmaster. Programs with *exec_permission* set to user can be accessed by users with UNIX accounts only. This option can be used with the `-a` and `-c` options. The directory from where this program is run as postmaster is the postmaster's home directory. If specified as `user`, then the user's home directory is the environment where the program is run as the user.
`-d`	Delete a method from the list of supported program delivery methods. This option cannot be used with the `-a`, `-c`, `-l`, or `-u` options.
`-h`	Help for this command.
`-l`	List all methods.
`-u`	List all users using the method specified with the `-m` option.

Examples

To add a method `procmail1` that executes the program `procmail` with the arguments `-d` *username* and executes as the user, enter the following:

```
imsimta program -a -m procmail1 -p procmail -g "-d %s" -e user
```

imsimta purge

The `imsimta purge` command deletes older versions of MTA log files. `imsimta purge` can determine the age of log files from the uniqueid strings terminating MTA log file names.

Syntax

```
imsimta purge [file-pattern] -day=dvalue -hour=hvalue -num=nvalue
```

Options

The options for this command are:

Option	Description
file-pattern	If specified, the *file-pattern* parameter is a file name pattern that establishes which MTA log files to purge. The default pattern, if none is specified, is `log/imta/log`.
`-day=`*dvalue*	Purges all but the last *dvalue* days worth of log files.
`-hour=`*hvalue*	Purges all but the last *hvalue* hours worth of log files.
`-num=`*nvalue*	Purges all but the last *nvalue* log files. The default is 5.

Example

To purge all but the last five versions of each type of log file in the `log/imta` directory:

```
imsimta purge
```

imsimta qclean

The `imsimta qclean` utility holds or deletes message files containing specific substrings in their envelope From:address, Subject: line, or content.

Syntax

```
imsimta qclean
  [-content=substring] [-from=substring] [-subject=substring]
  [-to=substring] [-domain_to=substring] [-database] [-delete | -hold]
  [-directory_tree] [-ignore_zz] [-match=keyword] [-min_length=n]
  [-threads | -nothreads] [-verbose | -noverbose] [channel]
```

Options

The options for this command are:

Option	Description
`-content=`*substring* `-from=`*substring* `-subject=`*substring* `-to=`*substring* `-domain_to=`*substring*	Specifies the substrings for which to search. Any combination of `-content`, `-from`, `-subject`, `-to`, and `-domain_to` may be specified. However, only one of each may be used. When a combination of such options is used, the `-match` option controls whether the options are interpreted as further restrictions (`-match=AND`) or as alternatives (`-match=OR`). The `-domain_to` option scans for frequently occurring envelope To: addresses. Identical to the `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.
`-database`	Specifies that only message files identified by the queue cache is searched.
`-delete`	Deletes matching message files.
`-hold`	Holds matching message files.
`-directory_tree`	Searches all message files that are actually present in the channel queue directory tree.
`-ignore_zz`	Ignores queued message files with file names beginning with “ZZ”. This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
`-match=`*keyword*	Controls whether a message file must contain all (`-match=AND`) or only one of (`-match=OR`) the specified substrings in order to be held or deleted. The default is `-match=AND`.

Option	Description
`-min_length=`*n*	Specifies the minimum length of the substring for which to search. By default, each substring must be at least 24 bytes long. Use the `-min_length` option to override this limit.
`-threads=`*n* | `-nothreads`	Accelerates the searching on multiprocessor systems by dividing the work amongst multiple, simultaneous running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be an integer between 1 and 8. The default is `-nothreads`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies an MTA channel area to be searched for matching messages. The * or ? wildcard characters may be used in the channel specification.

imsimta qm

The `imsimta qm` utility inspects and manipulates the channel queue directories and the messages contained in the queues. `imsimta qm` contains some functionality overlap with the `imsimta cache` and `imsimta counters` commands.

For example, some of the information returned by `imsimta cache -view` is also available through the `imsimta qm directory` command. However, `imsimta qm`, does not completely replace `imsimta cache` or `imsimta queue`.

You must be `root` or `mailsrv` to run `imsimta qm`.

`imsimta qm` can be run in an interactive or non-interactive mode. To run `imsimta qm` in the interactive mode, enter:

```
imsimta qm
```

You can then enter the sub-commands that are available for use in the interactive mode. To exit out of the interactive mode, enter `exit` or `quit`.

To run `imsimta qm` in the non-interactive mode, enter:

```
imsimta qm sub-commands [options]
```

Note that some of the sub-commands available in the interactive mode are not available in the non-interactive mode, and vice versa. See "Sub-Commands" on page 95 for descriptions of all available sub-commands. Each sub-command indicates the mode for which mode it is available.

Sub-Commands

clean

The clean sub-command holds or deletes message files containing specific substrings in their envelope From: address, Subject: line, or content.

Available in both interactive and non-interactive modes.

```
clean [-content=substring] [-from=substring] [-subject=substring]
  [-to=substring] [-domain_to=substring]
  [-database | -directory_tree] [-delete | -hold] [-ignore_zz]
  [-match=keyword] [-min_length=n] [-threads=n | -nothreads]
  [-verbose | -noverbose] [channel]
```

The options for this sub-command are:

Option	Description
-content=*substring* -from=*substring* -subject=*substring* -to=*substring* -domain_to=*substring*	Specifies the substrings for which to search. Any combination of each option may be used. However, only one of each may only be used. When a combination of such options is used, the -match option controls whether the options are interpreted as further restrictions (-match=AND), or as alternatives (-match=OR). The -domain_to option scans for frequently occurring envelope To: addresses. Identical to the -to option, except -domain_to looks at only the *host.domain* portion of the envelope To: address. The -from option can take an empty address using, for example, imsimta qm clean -from=\<\>.
-database | -directory_tree	Controls whether the message files searched are only those with entries in the queue cache (-database) or all message files actually present in the channel queue directory tree (-directory_tree). When neither -database nor -directory_tree is specified, then the view selected with the view sub-command will be used. If no view sub-command has been issued, then -directory_tree is assumed.
-delete | -hold	Specifies whether matching message files are held (-hold) or deleted (-delete). The -hold option is the default.

Option	Description
-ignore_zz	Ignores queued message files with file names beginning with “ZZ”. This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
-match=*keyword*	Controls whether a message file must contain all (-match=AND) or only one of (-match=OR) the specified substrings in order to be held or deleted. The substrings are specified by the -content, -env_from, and -subject options. The default is -match=AND.
-min_length=*n*	Overrides the length limit for each substring to be searched. By default, the limit is 24 bytes (-min_length=24).
-threads=*n* | -nothreads	Accelerates the searching on multiprocessor systems by dividing the work amongst multiple, simultaneous running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer between 1 and 8. The default is -nothreads.
-verbose | -noverbose	Requests that the utility displays operation information (-verbose). The default is -noverbose.
channel	Specifies a specific MTA channel area to be searched for matching messages. The * or ? wildcard characters may be used in the channel specification.

counters clear

The counters clear sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and association counters if the segment does not already exist.
2. Sets all counter values to zero.
3. When -channels is specified, sets the counts of stored messages, recipients, and volume from the queue cache database.

Available for both interactive and non-interactive modes.

```
counters clear [-channels] [-associations]
```

The options for this sub-command are:

Option	Description
-channels	Clears the message counters

Option	Description
`-associations`	Clears the association counters

When neither option is specified, both are assumed. When `-associations` is specified and `-channels` is not specified, step (3) above is not performed.

counters create

The `counters create` sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and association counters if the segment does not already exist.
2. Sets the counts of stored messages, recipients, and volume from the queue cache database.

Available for both interactive and non-interactive modes.

```
counters create [-max_channels=n]
```

The option for this sub-command is:

Option	Description
`-max_channels=`*n*	Tells the MTA how many channels to allow for in the memory segment. If this option is omitted, then the MTA looks at the `imta.cnf` file and determines a value on its own.

counters delete

The `counters delete` sub-command deletes the shared memory segment used for channel message and association counters. Note that active MTA server processes and channels will likely recreate the memory segment.

Available for both interactive and non-interactive modes.

```
counters delete
```

counters show

Use the `counters show` sub-command to display channel message counters. When the optional *channel-name* parameter is omitted, * (wildcard) is assumed and the message counters for all channels are displayed. The *channel-name* parameter may contain the * and? wildcard characters.

The *counters show* sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and associated counters if the segment does not already exist.
2. Sets the counts of stored messages, recipients, and volume from the queue cache database.
3. Displays the message counters for the specified channels.

Available for both interactive and non-interactive modes.

```
counters show [-headers] [-noheaders] [-output=file-spec] \
[channel-name]
```

The options for this sub-command are:

Option	Description
`-headers` or `-noheaders`	Controls whether or not a heading is displayed. The `-headers` option is the default.
`-output=`*file_spec*	Causes the output to be written to a file. Any existing file with the same name as the output file is overwritten.

date

Displays the current time and date in RFC 822, 1123 format.

Available for both interactive and non-interactive modes.

```
date
```

delete

Deletes the specified messages displayed in the most recently generated message queue listing.

```
delete [-channel=name [-all]]  [-confirm | -noconfirm]
  [-log | -nolog] [id...]
```

The *id* parameter specifies the messages to be deleted.

See "imsimta qm Options" on page 108 for information on using the -channel, -all, -confirm, and -log options.

Available only in interactive mode.

directory

Generates a listing of queued message files. By default, the imta/queue directory tree is used as the source of queued message information; this default may be changed with the view sub-command. The -database and -directory_tree options may be used to override the default.

Available for both interactive and non-interactive modes.

```
directory [-held | -noheld]  [-database]  [-directory_tree]
  [-envelope]  [-owner=username] [-from=address] [-to=address]
  [-match=bool] [-file_info | -nofile_info] [-total | -nototal]
  [channel-name]
```

The options for this sub-command are:

Option	Description
-database	Obtains message information from the Job Controller.
-directory_tree	Selects the on-disk directory tree as the source of message information.
-envelope	Generates a listing which also contains envelope address information.
-total|-nototal	Generates size and count totals across all selected channels.

Option	Description
-owner=*username*	Lists only those messages owned by a particular user. Messages enqueued by a local user will be owned by that user; most other messages will be owned by mailsrv. Use of the -owner option implies -database.
-from=*address* and -to=*address* and -match=*bool*	Lists only those messages with envelope From: or To: addresses matching the specified address. When both -from and -to are specified, a message is listed if either its envelope From: or To: addresses match the specified addresses. This corresponds to the -match=or option. Specify -match=and to list only messages matching both the specified From: and To: addresses. Use of -from or -to implies -envelope. *address* can include a wild card (*) that matches a sequence of characters or a % character that matches a single character.
-held | -noheld	By default, active messages are listed. Specify -held to instead list messages which have been marked as held. Note that -held implies -directory_tree.
-file_info | -nofile_info	When the directory tree is scanned, each message file is accessed to determine its size as measured in units of blocks (normally 1024 bytes). To suppress this behavior and speed up generation of the listing, specify -nofile_info. When the queue cache database is used, the -nofile_info option is ignored as the size information is stored in the database.
channel-name	Restricts the listing to one or more channels. If the *channel-name* parameter is omitted, a listing is made for all channels. The channel name parameter may contain the * and ? wildcard characters.

exit

Exits the imsimta qm utility. Synonymous with the quit sub-command.

Available for both interactive and non-interactive modes.

```
exit
```

held

Generates a listing of message files which have been marked as held. This listing is always generated from the imta/queue/ directory tree.

Available for both interactive and non-interactive modes.

```
held [-envelope] [-file_info | -nofile_info]  [-total | -nototal]
  [-from=address] [-to=address] [-match=bool] [channel-name]
```

The options for this sub-command are:

Option	Description
`-envelope`	Generates a listing which also contains envelope address information.
`-total` | `-nototal`	Generate size and count totals across all selected channels.
`-from=`*address* and `-to=`*address* and `-match=`*bool*	Lists only those messages with envelope From: or To: addresses matching the specified address. When both `-from` and `-to` are specified, a message is listed if either its envelope From: or To: addresses match the specified addresses. This corresponds to the `-match=or` option. Specify `-match=and` to list only messages matching both the specified From: and To: addresses. Use of `-from` or `-to` implies `-envelope`.
`-file_info` | `-nofile_info`	When the directory tree is scanned, each message file is opened to determine its size as measured in units of blocks (normally 1024 bytes). To suppress this behavior and speed up generation of the listing, specify `-nofile_info`.
channel-name	Restricts the listing to one or more channels. If the *channel-name* parameter is omitted, a listing is made for all channels. The *channel-name* parameter may contain the * and ? wildcard characters.

history

Displays any delivery history information for the specified messages from the most recently generated message queue listing.

Available only in interactive mode.

```
history [-channel=name [-all]  ]  [-confirm | -noconfirm] [id...]
```

Use the *id* parameter to specify the messages whose history is displayed.

See "imsimta qm Options" on page 108 for information on using the `-channel`, `-all`, and `-confirm` options.

hold

Marks as held the specified messages from the most recently generated message queue listing

Available only in interactive mode.

```
hold [-channel=name [-all]] [-confirm | -noconfirm]
  [-log | -nolog] [id...]
```

Use the *id* parameter to specify the messages to mark as held.

See "imsimta qm Options" on page 108 for information on the `-channel`, `-all`, `-confirm`, and `-log` options.

messages

The `imsimta qm messages` utility displays the number of messages queued for the channel given. Separate counts are given for messages that are ready to be processes (or are being processed) and those messages that have been tried and are awaiting their retry backoff. For channels where the destination host is significant, in particular the `tcp_*` channels, the messages are displayed by destination host.

```
messages channel
```

Example:

```
imsimta qm messages tcp_local
host                active messages   delayed messages
siroe.com              32                  47
west.siroe.com          0                   3
```

quit

Exits the `imsimta qm` utility. Synonymous with the `exit` sub-command.

Available in both interactive and non-interactive modes.

```
quit
```

read

Displays the specified messages from the most recently generated message queue listing.

Available only in interactive mode.

```
read [-content | -nocontent ]  [-channel=name  [-all]]
  [-confirm | -noconfirm] [id...]
```

The options for this sub-command are:

Option	Description
`-content` | `-nocontent`	Displays (`-content`) or suppresses display (`-nocontent`) of message content along with the envelope and header information. `-nocontent` is the default.
id	Specifies the messages to display.

See "imsimta qm Options" on page 108 for information on using the `-channel`, `-all`, and `-confirm` options.

release

If the specified message file is marked as held, it is renamed to remove the hold mark. The Job Controller, if running, is informed that the message is to be processed immediately, ahead of all other messages.

Available only in interactive mode.

```
release [-channel=name [-all]]  [-confirm | -noconfirm]
  [-log | -nolog] [id...]
```

Use the *id* parameter to specify the messages to release from `.HELD` status.

See "imsimta qm Options" on page 108 for information on using the `-channel`, `-all`, `-confirm`, and `-log` options.

return

Returns as undelivered the specified messages shown in the most recently generated message queue listing.

Available only in interactive mode.

```
return [-channel=name [-all]]  [-confirm  | -noconfirm]
  [-log | -nolog] [id...]
```

Use the *id* parameter to specify the messages to return.

See "imsimta qm Options" on page 108 for information on using the `-channel`, `-all`, `-confirm`, and `-log` options.

run

Processes, line-by-line, the commands specified in a file.

Available in both interactive and non-interactive modes.

```
run [-ignore | -noignore] [-log | -nolog] file-spec
```

Specifically, *file-spec* is opened and each line from it is read and executed.

The options for this sub-command are:

Option	Description
`-ignore` | `-noignore`	Unless `-ignore` is specified, command execution will be aborted should one of the sub-commands generate an error.
`-log` | `-nolog`	By default, each command is echoed to the terminal before being executed (the `-log` option). Specify `-nolog` to suppress this echo.

start

Restart processing of messages enqueued for the specified channel. The Job Controller not only marks the channel as “okay” to process, but it additionally starts processing jobs for the channel. This command takes effect whether the Job Controller is running or not.

```
start channel
```

The *channel* parameter specifies the channel to restart.

stop

Stops processing of messages enqueued for the specified channel. This command prevents you from having to stop the Job Controller and recompiling the configuration. The channel does not process messages until a `start` command is issued for that channel. This state persists across restarts of the Job Controller, the Messaging Server, and the host computer itself. This command takes effect whether the Job Controller is running or not.

```
stop channel
```

The *channel* parameter specifies the channel to stop.

summarize

The `summarize` sub-command displays a summary listing of message files.

```
summarize [-database | -directory_tree] [-heading | -noheading]
  [-held | -noheld] [-trailing | -notrailing]
```

This is a potentially expensive command if you have a large backlog of messages. Instead of running this command frequently, consider using the `imsimta qm messages` command.

The options for this sub-command are:

Option	Description
`-database` | `-directory_tree`	Controls whether the information presented is obtained from the Job Controller (`-database`) or by looking at the actual directory tree containing the channel queues (`-directory_tree`). When neither `-database` nor `-directory_tree` is specified, then the "view" selected with the `view` sub-command will be used. If no `view` sub-command has been issued, then `-directory_tree` is assumed.
`-heading` | `-noheading`	Controls whether or not a heading line describing each column of output is displayed at the start of the summary listing. The `-heading` option is the default.
`-held` | `-noheld`	Controls whether or not to include counts of .HELD messages in the output. The `-noheld` option is the default.
`-trailing` | `-notrailing`	Controls whether or not a trailing line with totals is displayed at the end of the summary. The `-trailing` option is the default.

top

The top sub-command displays the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. When used in conjunction with the `clean` sub-command, `top` may be used to locate unsolicited bulk email in the query and hold or delete it.

```
top [-content[=range]] [-from[=range]] [-subject[=range]]
  [-to[=range]] [-database | -directory_tree] [-domain_to[=range]]
  [-held] [-ignore_zz] [-min_count=n] [-threads=n | -nothreads]
  [-top=n] [-verbose | -noverbose] [channel]
```

The options for this sub-command are:

Option	Description
-content[=*range*] -from[=*range*] -subject[=*range*] -to[=*range*] -domain_to[=*range*]	The -content, -from, -subject, and -to options are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (-subject). Use -from to display frequent envelope From: fields, -to to display frequent envelope To: fields, or -content to display frequent message contents. Use -domain_to to display frequently occurring envelope To: addresses. Identical to -to option, except -domain_to looks at only the *host.domain* portion of the envelope To: address. Any combination of -content, -from, -to, -domain_to, and -subject may be specified. However, only one of each may be used. The -content, -from, -to, -domain_to, and -subject options accept the optional parameters START=*n* and LENGTH=*n*. These parameters indicate the starting position and number of bytes in the field to consider. The defaults are -content=(START=1,LENGTH=256), -from=(START=1,LENGTH=2147483647), -to=(START=1,LENGTH=2147483647), -subject=(START=1,LENGTH=2147483647), and -domain_to=(START=1,LENGTH=214783647). Use of these parameters is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.
-database | -directory_tree	Controls whether the message files scanned are only those with entries in the queue cache database (-database) or all message files actually present in the channel queue directory tree (-directory_tree). When neither -database nor -directory_tree is specified, then the "view" selected with the view sub-command will be used. If no view sub-command has been issued, then -directory_tree is assumed.
-held	Lists only the files which have a .HELD extension.
-ignore_zz	Ignores queued message files with file names beginning with "ZZ". This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
-min_count=*n*	Changes the minimum number of times that a string must occur in order to be displayed. The default is -min_count=2.
-threads=*n* | -nothreads	Accelerates searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer between 1 and 8. The default is -nothreads.
-top=*n*	Changes the amount of most frequently occurring fields that are displayed. The default is -top=20.

Option	Description
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies an MTA channel area to be scanned for string frequencies. The * or? wildcard characters may be used in the channel specification.

view

Specifies the source of queued message information for subsequent directory commands.

Available only in interactive mode.

```
view -database | -directory_tree
```

By default, queued message listings are generated by scanning the `imta/queue/` directory tree. This corresponds to the `-directory_tree` option. You can, alternatively, generate the listings from the MTA queue cache database by issuing the `-database` option.

Settings made with the `view` sub-command remain the default until either another view command is issued or the utility exits. The default may be overridden with the `-database` or `-directory_tree` options of the directory command.

Note that the directory tree is always used when listing held message files.

imsimta qm Options

The `delete`, `history`, `hold`, `read`, `release`, and `return` sub-commands all support the following options and parameter:

Option	Description
`-channel=`*name*	Operates on the specified channel.
`-all`	The `-all` option may be used to operate on all of the previously listed messages. When used in conjunction with the `-channel` option, only those previously listed messages for the specified channel are operated on. The `-all` option may not be used in conjunction with an *id* parameter. However, `-all` or at least one *id* parameter must be specified.

Option	Description
`-confirm` and `-noconfirm`	When the *id* parameter is not used to explicitly select messages, you will be prompted to confirm the operation. This prevents accidental `delete -all` sub-commands from being executed. You can use the `-noconfirm` option to suppress this prompt. Similarly, `-confirm` causes a confirmation prompt to be required.
`-log` and `-nolog`	Controls whether or not the operation on each selected message is reported.
id	The identification number of a message shown in the most recent listing generated by either the `directory` or the `held` sub-command. The identification number for a message is the integer value displayed in the left-most column of the listing. The *id* can also be a range or comma-separated list.

These options identify the messages to which the command is applied. When none of the options are specified, at least one *id* parameter must be supplied.

For example, in the following listing the first message displayed has an identification number of 1 and the second 2:

```
qm.maint> directory tcp_local

Channel: tcp_local                Size Queued since
-------------------------------------------------------------
1 XS01IVX1T0QZ18984YIW.00       24 16-APR-1998 00:30:30.07
2 YH01IW2MZLN0RE984VUK.00       24 20-APR-1998 00:30:40.31
```

These two messages can therefore be selected by either “1,2” or “1-2”.

Examples

Non-Interactive Mode

The following example generates a list of queued messages:

```
imsimta qm directory

Wed, 24 Feb 1999 14:20:29 -0800 (PST)
Data gathered from the queue directory tree

Channel: sims-ms                  Size Queued since
---------------------------------------------------------------
1 ZZ0F7O00I03CJHZD.00            1 24-Feb-1999 11:52:29
2 ZZ0F7O00I03CILY6.00            1 24-Feb-1999 11:51:57
---------------------------------------------------------------
Total size:                         2

Grand total size:                   2
```

Interactive Mode

In the following interactive session, the `directory` sub-command is used to obtain a list of queued messages. The `delete` sub-command is then used to delete the first of the displayed messages. Finally, another `directory` sub-command is issued that displays that the deleted message is indeed gone.

```
imsimta qm

qm.maint> directory

Thu, 25 Feb 1999 11:37:00 -0800 (PST)
Data gathered from the queue directory tree

Channel: sims-ms                  Size Queued since
------------------------------------------------------------
1 ZZ0F7O00I03CJHZD.00            1 24-Feb-1999 11:52:29
2 ZZ0F7O00I03CILY6.00            1 24-Feb-1999 11:51:57
------------------------------------------------------------
Total size:                          2

Grand total size:                    2

qm.maint> delete 1
%QM-I-DELETED, deleted the message file
msg-tango/imta/queue/sims-ms/013/ZZ0F7O00I03CJHZD.00

qm.maint> directory
Thu, 25 Feb 1999 11:37:09 -0800 (PST)
Data gathered from the queue directory tree

Channel: sims-ms                  Size Queued since
------------------------------------------------------------
1 ZZ0F7O00I03CILY6.00            1 24-Feb-1999 11:51:57
------------------------------------------------------------
Total size:                          1

Grand total size:                    1
```

imsimta qtop

The imsimta qtop utility displays the most frequently occurring envelope From:, To:, Subject:, or message content fields found in message files in the channel queues.

Syntax

```
imsimta qtop [-content[=range]] [-from[=range]] [-subject[=range]]
  [-to[=range]] [-domain_to[=range]] [-database | -directory_tree]
  [-ignore_zz] [-min_count=n] [-threads=n | -nothreads] [-top=n]
  [-verbose | -noverbose] [channel]
```

Options

The options for this command are:

Option	Description

`-content[=`*range*`]`
`-from[=`*range*`]`
`-subject[=`*range*`]`
`-to[=`*range*`]`
`-domain_to[=`*range*`]`
	Specifies which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Specify `-from` to display frequent envelope From: fields, `-to` to display frequent envelope To: fields, or `-content` to display frequent message contents. Specify `-domain_to` to display frequently occurring envelope To: fields. Identical to `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.
	
	Any combination may be specified. However, only one of each my be used. These options accept the `START=`*n* and `LENGTH=`*n* arguments. These arguments indicate the starting offset and number of bytes in the field to consider. The defaults are `-content=(START=1,LENGTH=256)`, `-from=(START=1,LENGTH=2147483647)`, `-subject=(START=1,LENGTH=2147483647)`, and `-domain_to=(START=1,LENGTH=2147483647)`.

`-database`	Specifies that only message files identified by the queue cache database is searched.
`-directory_tree`	Searches all message files actually present in the channel queue directory tree.
-ignore_zz	Ignores queued message files with file name beginning with “ZZ”. This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt. For example, the following command indicates to which domains the MTA has problems delivering messages: `imsimta qtop -ignore_zz -domain_to`
`-min_count=`*n*	Changes the minimum number of times that a string must occur in order to be displayed. The default is `-min_count=2`.

Option	Description
`-threads=`*n* | `-nothreads`	Accelerates searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be an integer between 1 and 8. The default is `-nothreads`.
`-top=`*n*	Changes the amount of most frequently occurring fields that are displayed. The default is `-top=20`.
`-verbose | -noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`
channel	Specifies a channel area to be scanned for string frequencies. The * and ? wildcard characters may be used in the channel specification.

imsimta refresh

The `imsimta refresh` utility performs the following functions:

- Recompiles the MTA configuration files.
- Stops any MTA Job Controller or MTA Service Dispatcher jobs that are currently running.
- Restarts the Job Controller and MTA Service Dispatcher.

Essentially, `imsimta refresh` combines the function of `imsimta cnbuild` and `imsimta restart`.

NOTE You must be logged in as `root` to run `imsimta refresh`.

Syntax

```
imsimta refresh [job_controller | dispatcher]
```

Options

The options for this command are:

Option	Description
`job_controller`	Restarts the Job Controller.
`dispatcher`	Restarts the MTA Service Dispatcher.

If no component name is specified, all active components are restarted.

imsimta reload

Some parts of the MTA configuration can be changed and have these changes activated without having to stop and start the system. The reloadable parts of the configuration are:

mappings

aliases

general, forward and reverse lookup tables

These can be changed, compiled, and the changes activated by issuing the commands:

```
imsimta cnbuild
```

```
imsimta reload
```

The imsimta reload command informs the dispatcher and job controller of the change, and they in turn inform the processes they started.

imsimta renamedb

The `imsimta renamedb` command renames an MTA database. Since the MTA may optionally reference several “live” databases, that is, databases whose presence triggers their use by the MTA, it is important, first, to ensure that the MTA does not see such a database while it is in a mixed state, and second, to minimize any period of time during which the database is inaccessible. The `imsimta crdb` command locks the database it is creating to avoid having it accessed in a mixed state.

It is therefore recommended that the MTA databases be created or updated in a two-step process:

1. Create or update a temporary database.
2. Rename the temporary database to the “live” name using the `imsimta renamedb` command.

The `imsimta renamedb` command, which must delete any old database files and rename the new database files, locks the database during the renaming process to avoid presenting the database in a mixed state. In this way the database is never accessible while it is in a mixed state, yet any window of time during which the database is inaccessible is minimized. Renaming is generally quicker than database generation.

Syntax

```
imsimta renamedb old-database-spec new-database-spec
```

Parameters

The parameters for this command are:

Parameter	Description
old-database-spec	The name of the database that is being renamed.
new-database-spec	The new name of the database. This may either be an actual pathname, or one of the special names such as `IMTA_ALIAS_DATABASE`, `IMTA_REVERSE_DATABASE`, `IMTA_GENERAL_DATABASE`, or `IMTA_DOMAIN_DATABASE`, listed in the MTA tailor file and pointing to actual pathnames.

Example

The following command renames the database `tmpdb` to be the actual MTA alias database (usually *msg_svr_base*`/data/db/aliasesdb`).

```
imsimta renamedb tmpdb IMTA_ALIAS_DATABASE
```

imsimta restart

The `imsimta restart` command stops and restarts the Job Controller and Service Dispatcher. This causes all MTA master and slave programs to be restarted. It can also restart SMTP, LMTP, and SMTP_SUBMIT.

Detached MTA processes should be restarted whenever the MTA configuration is altered—these processes load information from the configuration only once and need to be restarted in order for configuration changes to become visible to them. In addition to general MTA configuration files, such as the imta.cnf file, some components, such as the MTA Service Dispatcher, have their own specific configuration files, for example, `dispatcher.cnf`, and should be restarted after changes to any of these files.

NOTE You must be logged in as root to use this utility.

Syntax

```
imsimta restart [job_controller | dispatcher|smtp|lmtp|smtp_submit]
```

Restarting the MTA Service Dispatcher effectively restarts all the service components it handles. If no component name is given, all active components are restarted.

Example

To restart the MTA Job Controller and channel master programs:

```
imsimta restart job_controller
```

imsimta return

The `imsimta return` command returns a message to the message's originator. The returned message a single multipart message with two parts. The first part explains the reason why the message is being returned. The text of the reason is contained in the file `return_bounce.txt` located in the *msg_svr_base*`/config/locale/C/LC_MESSAGES` directory. The second part of the returned message contains the original message.

Syntax

```
imsimta return message-file
```

message-file is the name of the message file to return. The name may include wildcards, but if so, the specification must be quoted.

Example

The following command causes the specified the message to be returned to its originators.

```
imsimta return /imta/queue/l/ZZ0FRW00A03G2EUS.00
```

imsimta run

The `imsimta run` command processes the messages in the channel specified by the channel parameter. Output during processing is displayed at your terminal, which makes your terminal unavailable for the duration of the operation of the utility. Refer also to the `imsimta submit` command which, unlike `imsimta run`, does not monopolize your terminal.

Note that a channel delivery program that is run using this command, unlike the `imsimta submit` command, attempts to deliver messages before any pending backoff delay has expired.

Syntax

```
imsimta run channel
```

Parameters

The parameter for this command is:

Parameter	Description
channel	Specifies the channel to be processed. This parameter is mandatory.

Example

Type the following command to process any messages in the tcp_local channel:

```
imsimta run tcp_local
```

imsimta start

The `imsimta start` command starts up detached MTA processes. If no component parameter is specified, then the MTA Job Controller and MTA Service Dispatcher are started. Starting the Service Dispatcher starts all services the Service Dispatcher is configured to handle, which usually includes the SMTP server.

The services handled by the MTA Service Dispatcher must be started by starting the MTA Service Dispatcher. Only services not being handled by the MTA Service Dispatcher can be individually started via the `imsimta start` command. The Service Dispatcher may be configured to handle various services, for example, the multithreaded SMTP server.

NOTE You must be logged in as root to use this utility.

Syntax

```
imsimta start [component]
```

If a component parameter is specified, then only detached processes associated with that component are started. The standard component names are:

- `dispatcher`—Multithreaded Service Dispatcher.
- `job_controller`—Schedules deliveries (dequeues messages).

Example

Use the following command to start the MTA Job Controller and MTA Service Dispatcher:

```
imsimta start
```

imsimta stop

The `imsimta stop` command shuts down the MTA Job Controller and the MTA Dispatcher. Shutting down the MTA Dispatcher shuts down all services (for example, SMTP) being handled by the Dispatcher. It can also be used to stop the SMTP, LMTP, SMTP_SUBMIT servers.

NOTE You must be logged in as root to use this utility.

Syntax

```
imsimta stop [dispatcher | job_controller|smtp|smtp_submit|lmtp]
```

Example

Use the following command to shut down the MTA jobs:

```
imsimta stop
```

imsimta submit

The `imsimta submit` command directs the Job Controller to fork a process to execute the messages queued to the channel specified by the channel parameter.

Syntax

```
imsimta submit [channel] [poll]
```

Parameters

The parameters for this command are:

Parameter	Description
channel	Specifies the channel to be processed. The default, if this parameter is not specified, is the local channel 1.
`poll`	If poll is specified, the channel program runs even if there are no messages queued to the channel for processing.

Example

Use the following command to process any messages in the `tcp_local` channel:

```
imsimta submit tcp_local
```

imsimta test

The `imsimta test` utilities perform tests on various areas of functionality of the MTA.

imsimta test -mapping

`imsimta test -mapping` tests the behavior of a mapping table in the mapping file. The result of mapping an input string will be output along with information about any meta characters specified in the output string.

If an input string is supplied on the command line, then only the result of mapping that input string will be output. If no input string is specified, `imsimta test -mapping` will enter a loop, prompting for an input string, mapping that string, and prompting again for another input string. `imsimta test -mapping` will exit when a CTRL-D is entered.

imsimta test -match

`imsimta test -match` tests a mapping pattern in order to test wildcard and global matching.

`imsimta test -match` prompts for a pattern and then for a target string to compare against the pattern. The output indicates whether or not the target string matched. If a match was made, the characters in the target string that matched each wildcard of the pattern is displayed. The `imsimta test -match` utility loops, prompting for input until the utility is exited with a CTRL-D.

imsimta test -rewrite

`imsimta test -rewrite` provides a test facility for examining the MTA's address rewriting and channel mapping process without actually sending a message. Various qualifiers can be used to control whether `imsimta test -rewrite` uses the configuration text files or the compiled configuration (if present), the amount of output produced, and so on.

If a test address is specified on the command line, `imsimta test -rewrite` applies the MTA address rewriting to that address, reports the results, and exits. If no test address is specified, `imsimta test -rewrite` enters a loop, prompting for an address, rewriting it, and prompting again for another address. `imsimta test -rewrite` exits when CTRL-D is entered.

When testing an email address corresponding to a restricted distribution list, `imsimta test -rewrite` uses as the posting address the return address of the local postmaster, which is usually postmaster@localhost unless specified by the MTA option RETURN_ADDRESS in the MTA Option file.

imsimta test -url

`imsimta test -url` tests an LDAP queury URL. Note that the LDAP server to query is controlled by the setting of the MTA option `LDAP_SERVER` in `local.conf`.

Syntax

```
imsimta test -rewrite [-alias_file=filename]
  [-channel | -nochannel
  [-check_expansions | -nocheck_expansions]
  [-configuration_file=filename ]  [-database=database_list]
  [-debug | -nodebug]  [-delivery_receipt | -nodelivery_receipt]
  [-destination_channel=channel] [-filter | -nofilter]
  [-from=address | -nofrom] [-image_file=filename | -noimage_file]
  [-input=input-file] [-local_alias=value | -nolocal_alias]
  [-mapping_file=file | -nomapping_file]
  [-option_file=filename | -nooption_file]  [-output=output-file]
  [-read_receipt | -noread_receipt] [-restricted=setting]
  [-source_channel=channel] [-noreprocess]
```

```
imsimta test -mapping [input_string]  [-debug | -nodebug]
  [-flags=chars | -noflags]
  [-image_file=filename | -noimage_file] [-mapping_file=filename]
  [-option_file=filename | -nooption_file] [-table=table-name] [address]
```

```
imsimta test -match
```

```
imsimta test -url [-debug | -nodebug] [ldap_url]
```

```
imsimta test -message=message-file -exp -mm [-block] [-input=input-file]
  [-output=output-file]
```

Options

The options for this command are:

Option	Description
address	Specifies the test address to be rewritten. If this option is omitted, then the command prompts for an address. Used with the `-rewrite` option.
input_string	The string to be matched in the left side of a mapping table. Used with the `-mapping` option.
ldap_url	The LDAP URL that `imsimta test -url` attempts to resolve.
`-alias_file=`*filename*	Specifies an alternate alias file for `imsimta test -rewrite` to use. `imsimta test -rewrite` normally consults the default alias file named by the `IMTA_ALIAS_FILE` option of the MTA tailor file, *msg_svr_base*`/config/imta_tailor`, during the rewriting process. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; any compiled configuration precludes reading any sort of alias file.
`-block`	Treats the entire input as a single sieve script. The default is to treat each line as a separate script.

Option	Description
-channel | -nochannel	Controls whether imsimta test -rewrite outputs detailed information regarding the channel an address matches (e.g., channel flags).
-check_expansions | -nocheck_expansions	Controls checking of alias address expansion. Normally the MTA considers the expansion of an alias to have been successful if any of the addresses to which the alias expands are legal. The -check_expansions option causes a much stricter policy to be applied: imsimta test -rewrite -check_expansions checks each expanded address in detail and reports a list of any addresses, expanded or otherwise, that fail to rewrite properly.
-configuration_file=*file*	Specifies an alternate file to use in place of the file named by IMTA_CONFIG_FILE. Normally, imsimta test -rewrite consults the default configuration file named by the IMTA_CONFIG_FILE option of the MTA tailor file, *msg_svr_base*/config/imta_tailor, during the rewriting process. This option has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of configuration file.
-database=*database-list*	Disables references to various databases or redirects the database paths to nonstandard locations. imsimta test -rewrite normally consults the usual MTA databases during its operation. The allowed list items are alias, noalias, domain, nodomain, general, nogeneral, reverse, and noreverse. The list items beginning with “no” disable use of the corresponding database. The remaining items require an associated value, which is taken to be the name of that database.
-debug | -nodebug	Enables the production of the additional, detailed explanations of the rewriting process. This option is disabled by default.
-delivery_receipt | -nodelivery_receipt	Sets the corresponding receipt request flags. These options can be useful when testing the handling of sent or received receipt requests when rewriting forwarded addresses or mailing lists.
-destination_channel=*channel*	Controls to which destination or target channel imsimta test -rewrite rewrites addresses. Some address rewriting is destination channel specific; imsimta test -rewrite normally pretends that its channel destination is the local channel l.
-exp	Evaluates expressions that can be used in sieve scripts.
-filter | -nofilter	Outputs any filters that are applied for the specified address.
-from=*address* | -nofrom	Controls what envelope From: address is used for access control probes when the -from option is specified. If *address* is omitted, the postmaster return address is used for such probes. If the -nofrom option is specified, the MTA uses an empty envelope From: address for access probes.

Option	Description
-flags=*chars* | -noflags	Specifies particular flags to be set during the mapping test when the -flags option is specified. For example, *chars* can be E (envelope), B (header/body), or I (message id) when testing a REVERSE mapping. Used with the -mapping option only.
-image_file=[*filename*] | -noimage_file	The -noimage_file option instructs the command to unconditionally ignore any previously compiled configuration and to read configuration information from the various text files instead. When the -image_file option is specified without an optional file name, the compiled configuration is loaded from the file named by the IMTA_CONFIG_DATA option into the MTA tailor file, *msg_svr_base*/config/imta_tailor, which is usually *msg_svr_base*/config/imta.cnf. If, instead, a file name is specified, then the compiled configuration is loaded from the specified file.
-input=*input-file*	Specifies a source for input. By default, imsimta test takes input from stdin.
-local_alias=*value* | -nolocal_alias	Controls the setting of an alias for the local host. The MTA supports multiple "identities" for the local host; the local host may have a different identity on each channel. This option may be used to set the local host alias to the specified value; appearances of the local host in rewritten addresses are replaced by this value.
-mapping_file=*file* | -nomapping_file	Instructs the command to use the specified mapping file rather than the default mapping file named by the IMTA_MAPPING_FILE option in the MTA tailor file, *msg_svr_base*/config/imta_tailor, which is usually the file named *msg_svr_base*/config/mappings. This option has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration precludes reading the mappings file. Use of the -nomapping_file option will prevent the IMTA_MAPPING_FILE file from being read in when there is no compiled configuration.
-message=*message-file*	Specifies the text file containing the message that is tested. The *message-file* must be an RFC 822 message only; it cannot be a queue file.
-mm	Tells imsimta test -exp to load the sieve-specific extensions to the expression interpreter. This includes all the sieve tests and actions such as header, address, envelope, discard, fileinto, and keep. Without -mm you cannot test sieves. The command to test sieves against a message is: imsimta test -expression -mm -message=*message*
-noreprocess	Tuns off the internal reprocessing flag that simulates the behavior of other components that operate without the reprocessing flag being set.

Option	Description
`-option_file=`*filename* | `-nooption_file`	Instructs the command to use the specified option file rather than the default option file named by the `IMTA_OPTION_FILE` option in the MTA tailor file, *msg_svr_base*`/config/imta_tailor`, which is usually the file *msg_svr_base*`/config/options.dat`. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of option file. Use of the `-nooption_file` option prevents the `IMTA_OPTION_FILE` file from being read in when there is no compiled configuration.
`-output=`*output-file*	Directs the output of `imsimta test`. By default, `imsimta test` writes output to `stdout`. This option only works if the `mailsrv` account has write access to the current working directory.
`-read_receipt` | `-noread_receipt`	Sets the corresponding receipt request flags. This option can be useful when testing the handling of receipt requests at the time of rewriting forwarded addresses or mailing lists.
`-restricted=`*setting*	Controls the setting of the restricted flag. By default, this flag has value 0. When set to 1, `-restricted=1`, the restricted flag is set on and addresses are rewritten using the restricted mailbox encoding format recommended by RFC 1137. This flag is used to force rewriting of address mailbox names in accordance with the RFC 1137 specifications.
`-source_channel=`*channel*	Controls which source channel is performing the rewriting. Some address rewriting is source channel-specific; `imsimta test -rewrite` normally assumes that the channel source for which it is rewriting is the local channel l.
`-table=`*table-name*	Specifies the name of the mapping table to test. If this option is not specified, then `imsimta test -mapping` prompts for the name of the table to use.

Example

This example shows typical output generated by `imsimta test -rewrite`. The most important piece of information generated by `imsimta test -rewrite` is displayed on the last few lines of the output, which shows the channel to which `imsimta test -rewrite` would submit a message with the specified test address and the form in which the test address would be rewritten for that channel. This output is invaluable when debugging configuration problems.

```
imsimta test -rewrite

Address: joe.blue
channel = l
channel description =
channel description =
channel flags #1 = BIDIRECTIONAL MULTIPLE IMMNONURGENT NOSERVICEALL
channel flags #2 = NOSMTP POSTHEADBODY HEADERINC NOEXPROUTE
channel flags #3  = LOGGING NOGREY NORESTRICTED
channel flags #4 = EIGHTNEGOTIATE NOHEADERTRIM NOHEADERREAD RULES
channel flags #5 =
channel flags #6 = LOCALUSER NOX_ENV_TO RECEIPTHEADER
channel flags #7 = ALLOWSWITCHCHANNEL NOREMOTEHOST DATEFOUR DAYOFWEEK
channel flags #8 = NODEFRAGMENT EXQUOTA REVERSE NOCONVERT_OCTET_STREAM
channel flags #9       = NOTHURMAN INTERPRETENCODING

text/plain charset def = (7) US-ASCII 5 (8) ISO-8859-1 51
channel envelope address type = SOURCEROUTE
channel header address type = SOURCEROUTE
channel official host  = mailserver.eng.alpha.com

  channel local alias    =

  channel queue name     =

  channel after param    =

  channel daemon name    =

  channel user name      =

  notices                =

  channel group ids      =
```

```
  header To: address     = joe.blue@mailserver.eng.alpha.com

  header From: address   = joe.blue@mailserver.eng.alpha.com

  envelope To: address   = joe.blue@mailserver.eng.alpha.com  (route
(mailserver.eng.alpha.com,mailserver.eng.alpha.com))

  envelope From: address = joe.blue@mailserver.eng.alpha.com

  name                   =

  mbox                   = joe.blue

Extracted address action list: joe.blue@mailserver.eng.alpha.com

Extracted 733 address action list: joe.blue@mailserver.eng.alpha.com

Expanded address:

  joe.blue@mailserver.eng.alpha.com

Submitted address list:

  ims-ms

    joe.blue@ims-ms-daemon (sims-ms-daemon) *NOTIFY FAILURES* *NOTIFY
DELAYS*



Submitted notifications list:



Address:

#
```

In the following example, the sample PAGER mapping is tested. The -mapping_file option is used to select the mapping file pager_table.sample instead of the default mapping file.

```
imsimta test -mapping -noimage_file \
 -mapping_file=msg_svr_base/config/pager_table.sample
```

In the following example, the sample mapping pattern $[ax1]*@*.xyz.com is tested for several sample target strings:

```
imsimta test -match

Pattern: $[ax1]*@*.xyz.com
 [  1S] cglob [1ax]
 [  2] "@"
 [ 3S] glob, req 46, reps 2
 [  4] "."
 [  5] "x"
 [  6] "y"
 [  7] "z"
 [  8] "."
 [  9] "c"
 [ 10] "o"
 [ 11] "m"
Target: xx11aa@sys1.xyz.com
Match.
0 - xx11aa
1 - sys1
Pattern: $[ax1]*@*.xyz.com
Target: 12a@node.xyz.com
No match.
Pattern: $[ax1]*@*.xyz.com
Target: 1xa@node.acme.com
Match.
0 - 1xa
1 - node
Pattern: ^D
%
```

imsimta version

The `imsimta version` command prints out the MTA version number, and displays the system's name, operating system release number and version, and hardware type.

Syntax

```
imsimta version
```

Example

To check the version of MTA you are running, execute the following command:

```
% imsimta version
```

imsimta view

The `imsimta view` utility displays log files.

Syntax

```
imsimta view file-pattern [-f offset-from-first] [-l offset-from-last]
```

Options

The options for this command are:

Option	Description
`-f=`*offset-from-first*	Displays the specified version of the log file (starting from 0). For example, to find the earliest (oldest) version of the file, specify `-f=0`. By default, `imsimta view` finds the most recent version of the log file.
`-l=`*offset-from-last*	Displays the last version of the specified file. For example, to display the most recent (newest) version of the file, specify `-l=0`. By default, `imsimta view` finds the most recent version of the file.
file-pattern	Specifies a filename pattern to view.

Chapter 3

Messaging Server Configuration

This chapter lists the configuration parameters for the Messaging Server. These parameters can be set via the `configutil` command. For a full description and syntax of the `configutil` command, see "configutil" on page 17.

For information about configuring the MTA, see Chapter 4, "MTA Configuration."

configutil Parameters

Table 3-1 configutil Parameters

Parameter	Description
`alarm.msgalarmnoticehost`	Machine to which you send warning messages. If not set, localhost will be used. If you are using LMTP, set this to the machine name of the LMTP host. Default: localhost
`alarm.msgalarmnoticeport`	The SMTP port to which to connect when sending alarm messages. Default: 25
`alarm.msgalarmnoticercpt`	Recipient of alarm notice. Default: `Postmaster@localhost`
`alarm.msgalarmnoticesender`	Address of sender of alarm. Default: `Postmaster@localhost`
`alarm.msgalarmnoticetemplate`	Message template. %s in the template is replaced with the following (in order): sender, recipient, alarm description, alarm instance, alarm current value and alarm summary text
`alarm.diskavail.msgalarmstat interval`	Interval in seconds between disk availability checks. Set to 0 to disable checks of disk usage. Default: 3600

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
alarm.diskavail.msgalarmthreshold	Percentage of disk space availability below which an alarm is sent. Default: 10
alarm.diskavail.msgalarmthreshold direction	Specifies whether the alarm is issued when disk space availability is below threshold (-1) or above it (1). Default: -1
alarm.diskavail.msgalarmwarninginterval	Interval in hours between subsequent repetition of disk availability alarms. Default: 24
alarm.diskavail.msgalarmdescription	Percentage mail partition diskspace available.
alarm.serverresponse.msgalarmdescription	Server response time in seconds.
alarm.serverresponse.msgalarmstatinterval	Checking interval (seconds). Set to 0 to disable checking of server response. Default: 600
alarm.serverresponse.msgalarmthreshold	If server response time in seconds exceeds this value, alarm issued. Default: 10
alarm.serverresponse.msgalarmthresholddirection	Specifies whether alarm is issued when server response time is greater than (1) or less than (-1) the threshold. Default: 1
alarm.serverresponse.msgalarmwarninginterval	Interval in hours between subsequent repetition of server response alarm. Default: 24
encryption.nsssl2	Default: no
encryption.nsssl2ciphers	Comma-delineated list of ciphers
encryption.nsssl3	Default: yes
encryption.nsssl3ciphers	Default: rsa_rc4_40_md5, rsa_rc2_40_md5, rsa_des_sha,rsa_rc4_128_md5, rsa_3des_sha
encryption.nsssl3sessiontimeout	Default: 0
encryption.nssslclientauth	Default: 0
encryption.nssslsessiontimeout	Default: 0
encryption.fortezza.nssslactivation	Default: off
encryption.rsa.nssslactivation	Default: on

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
encryption.rsa.nssslpersonalityssl	Default: Server-Cert
encryption.rsa.nsssltoken	Default: internal
gen.accounturl	Location of the server administration resource for end users. Default: http://%U@[Hostname]:[AdminPort]/bin/user/admin/bin/enduser
gen.configversion	Configuration version. Default: 4.0.
gen.filterurl	URL for incoming mail (server side) filter.
gen.folderurl	URL for personal folder management.
gen.installedlanguages	Default: en
gen.listurl	URL for mailing list management.
gen.newuserforms	Welcome message for new users.The maximum size is 1 MB.
gen.sitelanguage	Default language tag. Default: en.
local.autorestart	Enable automatic restart of failed or frozen (unresponsive) servers including IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. Default: Off
local.autorestart.timeout	Failure retry time-out. If a server fails more than twice during this designated period of time, then the system will stop trying to restart this server. If this happens in 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. (See ocal.schedule.msprobe.) Default: 600 seconds
local.cgiexeclist	List of pattern string used to match command to be executed.
local.dbstat.captureinterval	Interval to capture db statistics into counters (seconds). Default: 3600.
local.defdomain	Default domain - set by install.
local.enablelastaccess	Enables imsconnutil to provide last log in information.
local.enduseradmincred	Password for end user administrator.
local.enduseradmindn	User id for end user administrator.
local.ens.enable	Enable ens server on start-msg startup. Default: On
local.ens.port	Set ENS (Event Notification Server) port and/or address. Syntax: configutil local.ens.port [*address*:]*port* Default: The ENS default

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.hostname`	Fully qualified DNS hostname of this mail server.
`,local.http.enableuserlist`	Enables `imsconnutil` on Messenger Express service.
`local.imap.enableuserlist`	Enables `imsconnutil` on IMAP service.
`local.imap.immediateflagupdate`	When set to yes, all changes to flags (message status) are updated in the database on disk immediately, instead of being buffered and updated once in a while. Default: no
`local.imta.enable`	Enable `imta` server on `start-msg` startup. Default: On
`local.imta.hostnamealiases`	Defines the list of hosts used to determine the local host name in direct LDAP lookups. This parameter can be overridden with the `LDAP_HOST_ALIAS_LIST` MTA option.
`local.imta.imta_tailor`	Location of the `imta_tailor` file for this MTA instance.
`local.imta.lookupandsync`	Defines which type of entries should be synched when using the direct LDAP lookup module. Specify 1 for users (default), 2 for groups, or 3 for users and groups.
`local.imta.lookupfallbackaddress`	When using the direct LDAP lookup module, this parameter allows the last alias lookup to be skipped. Instead the recipient address is rewritten to a fixed address. This parameter is used in conjunction with a `SEND_ACCESS` mapping rule to return an error code.
`local.imta.lookupmaxnbfailed`	The MTA does not honor this parameter.
`local.imta.mailaliases`	List of comma-delineated LDAP attributes that override the default attributes. These attributes should be email addresses that can be routed. For example: if `local.imta.mailaliases=`*mail,mailAlternateAddress,rfc822mailbox, rfc822mail alias*, the MTA will consider these attributes when routing messages. Default: *mailAlternateAddress*
`local.imta.schematag`	Defines the types of LDAP entries that are supported by the MTA. Default: ims50.
`local.imta.reverseenabled`	Triggers the generation of the reverse database. How the reverse database is actually used is controlled by the `USE_REVERSE_DATABASE` option. Default: yes.
`local.imta.catchallenabled`	Controls whether or not catch all addresses (mail or *mailAlternateAddress* in the form `@domain`) are enabled. Default: yes.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.imta.ssrenabled`	Triggers the generation of the server side rule database. How the SSR database is actually used is controlled by the `ssr` channel keyword. Default: yes
`local.installeddir`	Full pathname of software installation directory.
`local.instancedir`	Full pathname of server instance directory.
`local.lastconfigfetch`	Last configuration fetch timestamp.
`local.ldapbasedn`	Root for the config tree in the config LDAP. The config LDAP is read by all the store processes when starting up.
`local.ldapcachefile`	Location of cached configuration.
`local.ldaphost`	DN in the configuration directory under which configuration information for a specific server is stored.
`local.ldapisiedn`	Installed software DN.
`local.ldappoolrefreshinterval`	Length of time in minutes before LDAP connections are automatically closed then re-established to the LDAP server. Also, length of elapsed time in minutes until the failover directory server reverts back to the primary directory server. Default: -1 (never refresh)
`local.ldapport`	LDAP port. Default: 389.
`local.ldapsiecred`	Server credential.
`local.ldapsiedn`	Server instance entry DN.
`local.ldapusessl`	Sets whether or not LDAP auth uses SSL. Default: no.
`local.mmp.enable`	Enable `mmp` server on `start-msg` startup. Default: On
`local.poplogmboxstat`	Pop log will show mailbox statistics on login and logout if the value is set to 1. Default: 0 (off)
`local.queuedir`	Full pathname of spool directory.
`local.report.reportercmd`	Command to run in order to generate reports. Default: *msg_svr_base*`/bin/msg/admin/bin/reporter.pl`
`local.report.runinterval`	Interval for job generation process to sleep in between checking for jobs (seconds). Default: 3600.
`local.report.counterlogfile.expirytime`	Maximum time (in seconds) a logfile is kept. Default: 604800.
`local.report.counterlogfile.interval`	The frequency that the counter is captured in seconds. Default: 600.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.report.counterlogfile.logdir`	Directory path for log files.
`local.report.counterlogfile.loglevel`	Default: Notice.
`local.report.counterlogfile.maxlogfiles`	Maximum number of files. Default: 10.
`local.report.counterlogfile.maxlogfilesize`	Maximum size (bytes) of each log file. Default: 2097152.
`local.report.counterlogfile.maxlogsize`	Maximum size of all logfiles. Default: 20971520
`local.report.counterlogfile.minfreediskspace`	Minimum amount of free disk space (bytes) that must be available for logging. Default: 5242880.
`local.report.counterlogfile.rollovertime`	The frequency in which to rotate logfiles (in seconds). Default: 86400.
`local.report.counterlogfile.separator`	Field separator in counter logfile. Default: '\t'.
`local.report.job.desc.sample`	Description for report job sample.
`local.report.job.range.sample`	Time range of input data.
`local.report.job.schedule.sample`	The time to start reporting process.
`local.report.job.target.sample`	Location to send the report.
`local.report.job.type.sample`	Type of report for this job. Default: listmbox.
`local.report.type.cmd.listmbox`	Command to execute listmbox report type.
`local.report.type.desc.listmbox`	Description for listmbox report type.
`local.rfc822header.fixcharset`	Character set where improperly encoded 8-bit message headers are interpreted by Messenger Express.
`local.rfc822header.fixlang`	Specifies two-letter language ID where improperly encoded 8-bit message headers are interpreted by Messenger Express. This parameter must be used in conjunction with the `fixcharset` parameter.
`local.sched.enable`	Enable `sched` server on `start-msg` startup. Default: On

Table 3-1 configutil Parameters *(Continued)*

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. Both will be required if specified. 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 * * * /sbin/imexpire`

`local.schedule.msprobe`	`msprobe` run schedule. `msprobe` is a daemon that probes servers to see if they respond to service requests. The value is a crontab-style schedule string (see `local.schedule.purge`). Default: 600 seconds
`local.schedule.purge`	Interval for running `purge`. Uses UNIX `crontab` format: *minute hour day-of-month month-of-year day-of-week*. See "`local.schedule.expire`" above. Default: `0 0,4,8,12,16,20 * * * /opt/SUNWmsgsr/lib/purge -num=5`
`local.schedule.return_job`	Interval for running the return_job. Uses UNIX crontab format: *minute hour day-of-month month-of-year day-of-week*. Default: `30 0 * * * /opt/SUNWmsgsr/lib/return_job`

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.schedule.`*taskname*	A command and a schedule on which to execute the command. Format: `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 "`local.schedule.expire`" 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* Default: `N/A`
`local.schedule.userpurge`	Schedules purging of users. It uses a crontab-style entry. *minute hour day-of-month month-of-year day-of-week* is the schedule for running the command. It follows the UNIX `crontab` format.
`local.servergid`	Server groupid in UNIX. Default: nobody.
`local.servername`	Server name.
`local.serverroot`	Server root.
`local.servertype`	Server type. Default: msg.
`local.serveruid`	User id of server in UNIX. Default: msgsrv.
`local.service.http.filterhiddenmailinglists`	Excludes the `mgmanhidden` attribute from the search filter when set to 0. Default: 1

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.service.http.gzip.dynamic`	Enables or disables compression of dynamic content (for example: request to *.msc. files) delivered to Messenger Express or Communications Express mail clients. This can be disabled if Messenger Express or Communications Express users are getting corrupted content and cannot open their mail pages. Default: 1 (enable)
`local.service.http.gzip.static`	Enables or disables compression of static content (for example: html files) delivered to Messenger Express or Communications Express mail clients. This can be disabled if Messenger Express or Communications Express users are getting corrupted content and cannot open their mail pages. Default: 1 (enable)
`local.service.http.maxcollectmsglen`	Maximum message size the server collects from a remote POP mailbox. If any message in the mailbox to be collect exceeds this size, the collection will halt when that message is encountered.
`local.service.http.maxldaplimit`	Sets the maximum LDAP lookup limit. Default: 500
`local.service.http.proxy`	Enables the Messenger Express Multiplexor on a Messaging Server proxy machine (when set to 1). This specialized server acts as a single point of connection to Messenger Express (the HTTP access service) when managing multiple mail servers. Default: 0
`local.service.http.proxy.port.`*hostname*	Configures the port number of the back-end Messenger Express (HTTP) server with the Messaging Multiplexor.
`local.service.http.rfc2231compliant`	Enables WebMail's RFC-2231 encoder so that the attachment filename will be encoded in the method defined by RFC-2231.
`local.service.http.smtpauthpassword`	Password for end user AUTH SMTP user.
`local.service.http.smtpauthuser`	User id for end user AUTH SMTP user. This parameter allows someone using Messenger Express to receive the same authenticated SMTP messages that they would normally receive using another web browser. In order for this to work, the user ID and password given to the `mshttpd` must be a store administrator; they must exist in the `store.admins` list (for example, `admin` and `admin`). After setting these parameters, any mail received from a local user should have the word "Internal" appearing next to the "From:" header in the Message View window.
`local.service.pab.alwaysusedefaulthost`	Enables one PAB server to be used. Default: False

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.service.pab.attributelist`	Add new attributes to a personal address book entry. With this parameter, you can create an attribute that does not already exist. Default: `pabattrs`.
`local.service.pab.enabled`	Enable or disable PAB feature. Default: 1
`local.service.pab.ldapbasedn`	Base DN for PAB searches. Default: `o=pab`
`local.service.pab.ldapbinddn`	Bind DN for PAB searches.
`local.service.pab.ldaphost`	Hostname where Directory Server for PAB resides.
`local.service.pab.ldappasswd`	Password for user specified by `local.service.pab.ldapbinddn`.
`local.service.pab.ldapport`	Port number of the PAB Directory Server.
`local.service.pab.maxnumberofentries`	Maximum number of entries a single PAB can store. Default: 500
`local.service.pab.migrate415`	Enables PAB migration when set to “on”. The default value is “`off`”.
`local.service.proxy.serverlist`	Message store server list. Takes a space-separated strings. Not configured by default
`local.service.proxy.admin`	Default store admin login name. Not configured by default
`local.service.proxy.adminpass`	Default store admin password. Not configured by default.
`local.service.proxy.admin.`*`hostname`*	Store admin login name for a specific host. Not configured by default.
`local.service.proxy.adminpass.`*`hostname`*	Store admin password for a specific host. Not configured by default.
`local.smsgateway.enable`	Enable `sms` server on `start-msg` startup. Default: On
`local.snmp.enable`	Enable `snmp` server on `start-msg` startup. Default: On
`local.store.expire.cleanonly`[1]	*For backward compatibility.* Perform `purge` only, do not perform `imexpire`. Default: false
`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

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`local.store.expire.workday`[1]	*For backward compatibility.* Perform expire/cleanup on this day of the week. Values: 0-6 (0= Sunday) Default: None
`local.store.notifyplugin.noneInbox.enable`	Determines whether all folders generate notifications or if only the INBOX generates notifications: 0 = changes to the INBOX (only the INBOX) generate event notifications. 1=changes to any and every folder generate event notifications.
`local.store.quotaoverdraft`	Allows overdraft of message store quota. Accepted values are “on” and “off.” Off - rejects messages that would push the message store over its quota. On - allows messages to be delivered to the users until the usage is over the quota limit, at which time messages are deferred or bounced, the quota warning message is sent, and the quota grace period timer starts. Default: off
`local.store.serversidewastebasket`	Enables server side wastebasket. Accepted values are: “yes” and “no.” Default: no
`local.store.sharedfolders`	Disables listing of sharedfolders with “*” as its pattern. You can still select the shared folder, but you cannot list it with a “*”. Default: on
`local.store.snapshotdirs`	Number of separate snapshots to store on disk. Minimum is 2. Recommend enough to be sure you have a good database back by the time you figure out the current one is beyond repair. Default: 3
`local.store.snapshotinterval`	Interval of time between snapshots. Unit of time is in minutes. It is recommended that you perform this procedure at least once a day. Default: 0.
`local.store.snapshotpath`	Specifies the path in which to copy the `mboxlist` directory. Permissions must be set for the message store owner. Snapshots will be placed in subdirectories.
`local.store.deadlock.autodetect`	Sets whether all or just one thread resolves deadlock. Default: no.
`local.store.deadlock.checkinterval`	Specifies the sleep length (in microseconds) before lock_detect is set again. Default: 1000.
`local.supportedlanguages`	Languages supported by server code.
`local.tmpdir`	Default value for `service.http.spooldir`.
`local.ugldapbasedn`	Root of the user/group configuration tree in the Directory Server.
`local.ugldapbindcred`	Password for the user/group administrator.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
local.ugldapbinddn	DN of the user/group administrator.
local.ugldaphasplainpasswords	Sets whether the user/groups LDAP server is configured to store user passwords in plaintext and readable to the server. Default: no.
local.ugldaphost	LDAP server for user lookup.
local.ugldapport	LDAP port. Default: 389.
local.ugldapuselocal	If set to yes, the ugldap config data will be stored in the local config file. Otherwise, it is stored in LDAP. Default: yes
local.ugldapusessl	Sets whether or not to use SSL to connect to LDAP server. Default: no.
local.watcher.enable	Enable watcher on start-msg startup. Watcher is a daemon that monitors Messaging Server and restarts services that fail. Refer to local.auto.restart and the *Sun Java System Messaging Server Administration Guide* for details Default: On
local.watcher.port	Watcher listen port. Default: 49994
local.webmail.sieve.port	The port of the web container where the Mail Filter has been deployed.
local.webmail.sso.cookiedomain	Specifies the value to include in the domain field of any SSO cookie that is sent back to the client.
local.webmail.sso.enable	Performs all SSO functions, including accepting and verifying SSO cookies presented by the client when the login page is fetched. It returns an SSO cookie to the client for a successful login and responds to requests from other SSO partners to verify its own cookies. If set to zero, the server does not perform any SSO functions. The default is 0. This parameter takes an integer value.
local.webmail.sso.id	Specifies the application ID value when formatting SSO cookies set by the WebMail server. The default is NULL. This parameter takes a string value.
local.webmail.sso.prefix	Specifies the prefix value when formatting SSO cookies set by the WebMail server. Only SSO cookies with this prefix value are recognized by the server; all other SSO cookies are ignored. The default is NULL. This parameter takes a string value.
local.webmail.sso.singlesignoff	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 0, the WebMail server only clears its own SSO cookie. The default is 0.
logfile.*.buffersize	Size of log buffers (in bytes). Default: 0. * can be one of the following components: admin, default, http, imap, imta, pop.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
logfile.*.expirytime	Amount of time logfile is kept (in seconds). Default: 604800. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.flushinterval	Time interval for flushing buffers to log files (in seconds). Default: 60. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.logdir	Directory path for log files. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.loglevel	* can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.logtype	* can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.maxlogfiles	Maximum number for files. Default: 10. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.maxlogfilesize	Maximum size (bytes) of each log file. Default: 2097152. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.maxlogsize	Maximum size of all logfiles. Default: 20971520. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.minfreediskspace	Minimum amount of free disk space (bytes) that must be available for logging. Default: 5242880. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.rollovertime	The frequency in which to rotate logfiles (in seconds). Default: 86400. * can be one of the following components: admin, default, http, imap, imta, pop.
logfile.*.syslogfacility	Specifies whether or not logging goes to syslog. * can be one of the following components: admin, default, http, imap, imta, pop. The values can be 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 other log file service options are ignored. Default: none (logging uses the Message Server log files).
logfiles.admin.alias	Default: |logfile|admin
logfiles.default.alias	Default: |logfile|default
logfiles.http.alias	Default: |logfile|http
logfiles.imap.alias	Default: |logfile|imap
logfiles.imta.alias	Default: |logfile|imta
logfiles.pop.alias	Default: |logfile|pop

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
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
sasl.default.ldap.has_plain_passwords	Boolean to indicate directory stores plaintext passwords which enables APOP, CRAM-MD5 and DIGEST-MD5. 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.
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.transition_criteria	No longer supported or used. See sasl.default.auto_transition.
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

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
service.authcachesize	The number of concurrent users/entries in the cache during the service.authcachettl time interval. The unit is in “entries” and each entry takes 60 bytes. Default: 10000
service.authcachettl	Cache entry TTL in seconds. Default: 900.
service.dcroot	Root of DC tree in Directory Server. Default: o=Internet.
service.defaultdomain	Used to determine whether the domain is the default domain or a hosted domain.
service.dnsresolveclient	Sets whether or not to reverse name lookup client host. Default: no.
service.http.allowadminproxy	Sets whether or not to allow admin to proxy auth. Default: no.
service.http.allowanonymouslogin	Sets whether or not to allow anonymous login. Default: no.
service.http.connlimits	Maximum number of connections per IP address. The syntax is: r*ealm1,realm2,...* where a realm has the form of address ranges and maximum number of connections expressed as: IP “|” MASK “:” NUM There should be at least 1 realm of the form: 0.0.0.0|0.0.0.0:n to cover the default case.
service.http.domainallowed	Access filters for HTTP services.
service.http.domainnotallowed	Deny filters for HTTP services.
service.http.enable	Enable http server on start-msg startup. Default: On
service.http.enablesslport	Sets whether or not the service is started on a sslport. If both service.http.enable and service.http.enablesslport are turned off, then stored does not try to monitor http. Default: yes.
service.http.extraldapattrs	Extra LDAP attributes for customization.
service.http.fullfromheader	Sets whether or not to send complete “from” header. Default: no.
service.http.idletimeout	Idle timeout (in minutes). Default: 3.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
service.http.ipsecurity	Sets whether or not to restrict session access to login IP addresses. If set to yes, when the user logs in, the server remembers which IP address the user used to log in. Then it only allows that IP address to use the session cookie it issues to the user. Default: yes.
service.http.maxmessagesize	Maximum message size client is allowed to send. Default: 5242880.
service.http.maxpostsize	Maximum http post content length. Default: 5242880.
service.http.maxsessions	Maximum number of sessions per server process. Default: 6000.
service.http.maxthreads	Maximum number of threads per server process. Default: 250.
service.http.numprocesses	Number of processes. Default: 1.
service.http.plaintextmincipher	Sets plain text login allowance. Specify 0 to allow plain text login always. Specify -1 to never allow plain text login. Specify 40 or 128 to require login using encryption using 40 or 128 bit key. Default: 0.
service.http.port	Server port number. Default: 80.
service.http.proxydomainallowed	Access filters for proxy authentication to the HTTP service.
service.http.resourcetimeout	Webmail resource reduction timeout (in seconds). Default: 900.
service.http.sessiontimeout	Webmail client session timeout in seconds. Default: 7200
service.http.smtphost	SMTP relay host. If you are using LMTP, set this to the machine name of the LMTP host. Default: localhost
service.http.smtpport	SMTP relay port. Default: 25.
service.http.sourceurl	Webmail server URL.
service.http.spooldir	Spool directory for outgoing client mail.
service.http.sslcachesize	Number of SSL sessions to be cached. Default: 0.
service.http.sslport	SSL server port number. Default: 443.
service.http.sslsourceurl	Webmail server URL.
service.http.sslusessl	Sets whether or not to enable SSL. Default: yes.
service.imap.allowanonymouslogin	Allows anonymous login. Default: no.
service.imap.banner	IMAP protocol welcome banner.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`service.imap.connlimits`	Maximum number of connections per IP address. The syntax is: r*ealm1,realm2,...* where a realm has the form of address ranges and maximum number of connections expressed as: IP "|" MASK ":" NUM There should be at least 1 realm of the form: 0.0.0.0|0.0.0.0:n to cover the default case.
`service.imap.domainallowed`	Access filters for IMAP services.
`service.imap.domainnotallowed`	Deny filters for IMAP services.
`service.imap.enable`	Enable `imap` server on `start-msg` startup. Default: On
`service.imap.enablesslport`	Sets whether or not service is started on sslport. Default: yes.
`service.imap.idletimeout`	Idle timeout (in minutes). Default: 30.
`service.imap.maxsessions`	Maximum number of sessions per server process. Default: 4000.
`service.imap.maxthreads`	Maximum number of threads per server process. Default: 250.
`service.imap.numprocesses`	Number of processes. Default: 1.
`service.imap.plaintextmincipher`	Sets plain text login allowance. Specify 0 to allow plain text login always. Specify -1 to never allow plain text login. Specify 40 or 128 to require login using encryption using 40 or 128 bit key. Default: 0.
`service.imap.port`	Server port number. Default: 143.
`service.imap.sslcachesize`	Number of SSL sessions to be cached. Default: 0.
`service.imap.sslport`	SSL server port number. Default: 993.
`service.imap.sslusessl`	Sets whether or not SSL is enabled. Default: yes.
`service.listenaddr`	The IP address on which to listen.
`service.loginseparator`	The character to be used as the login separator. Default: @.
`service.plaintextloginpause`	The pause interval after successful login. Default: 0.
`service.pop.allowanonymouslogin`	Sets whether or not anonymous login is allowed. Default: no.
`service.pop.banner`	POP protocol welcome banner.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
service.pop.connlimits	Maximum number of connections per IP address. The syntax is: r*ealm1,realm2*,... where a realm has the form of address ranges and maximum number of connections expressed as: IP "|" MASK ":" NUM There should be at least 1 realm of the form: 0.0.0.0|0.0.0.0:n to cover the default case.
service.pop.domainallowed	Access filters for POP services.
service.pop.domainnotallowed	Deny filters for POP services.
service.pop.enable	Enable pop server on start-msg startup. Default: On
service.pop.enablesslport	Sets whether or not service is started on sslport. Default: yes.
service.pop.idletimeout	Idle timeout (in minutes). Default: 10.
service.pop.maxsessions	Maximum number of sessions per server process. Default: 600.
service.pop.maxthreads	Maximum number of threads per server process. Default: 250.
service.pop.numprocesses	Number of processes.
service.pop.plaintextminciph er	Sets plain text login allowance. Specify 0 to allow plain text login always. Specify -1 to never allow plain text login. Specify 40 or 128 to require login using encryption using 40 or 128 bit key. Default: 0.
service.pop.port	POP server port number. Default: 110.
service.pop.sslport	SSL server port number. Default: 992.
service.pop.sslusessl	Sets whether or not to enable SSL. Default: yes.
service.readtimeout	Period that msprobe waits after sending an request that goes unfulfilled before restarting a service. See local.schedule.msprobe. Default: 10 seconds
service.sslpasswdfile	Password for each keyfile.
store.admins	Space separated list of user ids with Message Store Administrator privileges.
store.cleanupage	Age (in hours) of expired or expunged message before purge will permanently remove it. Default: None
store.dbcachesize	Mailbox list database cache size. Default: 8388608

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`store.dbtmpdir`	The “mailbox list database temporary directory” - defined by the store.dbtmpdir configutil parameter - is a directory which is very heavily accessed. At install time, the value of this parameter is not defined and defaults to a subdirectory underneath the *msg_svr_base* location. If the disks that house the mboxlist database temporary directory are not fast enough at very large sites, performance problems might occur. As part of their performance and tuning steps, sites should take a note of this and define a value for this parameter which either points to a memory mapped file system, or which points to a location on a fast file system.
`store.defaultacl`	Default ACL.
`store.defaultmailboxquota`	Default mailbox quota, if not specified in user account. The mailbox quota is the total size of the mailbox in bytes. Accepts an integer value. Default: -1 (infinite).
`store.defaultmessagequota`	Default message quota, if not specified in user account. The message quota is the number of messages. Accepts an integer value. Default: -1 (infinite).
`store.defaultpartition`	Default partition.
`store.diskflushinterval`	Default: 15

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`store.expirerule.`*`rulename.attribute`*	A criteria for an `imexpire` rule. A rule consists of a set of attributes including a folder pattern, an exclusive flag, and one or more expiration criteria. Attributes (defaults): `exclusive` (yes/no). Specifies if this an exclusive rule. `folderpattern` (POSIX regular expression). The folders affected by this rule. `messagecount` (integer). Number of messages in a folder. `foldersize` (integer in bytes). Size of an over-sized message. `messagedays` (integer in days). Days that a message should remain in a folder. `messagesize` (integer in bytes). Size of an over-sized message. `messagesizedays` (integer in days). Days an over-sized message should remain in a folder *message header field* (string). Any field in a message header. `seen` (and/or). *Seen* is a message status flag. This attribute set to `and` specifies that the message must be seen **and** other criteria must be met before the rule is fulfilled. Set to `or`, this attribute specifies that the message only need to be seen **or** another criteria be met before the rule is fulfilled. `deleted` (and/or). *Deleted* is a message status flag. This attribute set to `and` specifies that the message must be seen **and** other criteria must be met before the rule is fulfilled. Set to `or`, this attribute specifies that the message only need to be seen **or** another criteria be met before the rule is fulfilled. Only one attribute per line can be specified. See the *Sun Java System Messaging Server Administration Guide* for details and examples. Default: Not applicable.
`store.expirerule.*.exclusive`	When this parameter is set to 'yes,' it is the only rule applied even if other rules match the given criteria. Default: no
`store.expirerule.*.folderpattern`	Folders by which the rules apply
`store.expirerule.*.foldersizebytes`	Maximum number of bytes in a folder.
`store.expirerule.*.messagecount`	Upper limit on number of messages to be kept in the specified folders.
`store.expirerule.*.messagedays`	Upper limit on how long a message is kept in the specified folders.
`store.expirerule.*.messagesize`	Maximum number of bytes in a message.

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`store.expirerule.*.messagesizedays`	Length of time messagesize message can stay.
`store.expirestart`[1]	*For backward compatibility.* `imexpire` start time. Format: 0-23 (represents hour) Default: None.
`store.partition.*.path`	Store partition directory path.
`store.partition.primary.path`	Full path name of the primary partition Default: *msg_svr_base*`/store/partition/primary`
`store.quotaenforcement`	Turns quotaenforcement on or off. Default: on.
`store.quotaexceededmsg`	Message to be sent to user when quota exceeds `store.quotawarn`. The message must contain a header (with at least a subject line), followed by $$, then the message body. The $ represents a new line. Depending on the shell that you are using, it might be necessary to append a \ before $ to escape the special meaning of $. ($ is often the escape character for the shell.) For example, to enable this parameter, you can set the following configuration variables: `configutil -o store.quotaexceededmsg -v` *'Subject: WARNING: User quota exceeded*$$*User quota threshold exceeded - reduce space used.'* `configutil -o store.quotanotification -v on` Default: null
`store.quotaexceededmsginterval`	Interval (in days) to wait before sending another quotaexceededmsg. Accepts an integer value. Default: 7.
`store.quotagraceperiod`	Time (in hours) the mailbox is over quota before the server starts bouncing the messages. Messages will remain in the queue until one of the following: • The mailbox no longer exceeds the quota. • The user has remained over quota longer than the specified grace period. • The message has remained in the queue longer than the maximum message queue time. Default: 120.
`store.quotanotification`	Enables quota notification for the Message store. Accepted values are “on” and “off”. Default: on

Table 3-1 configutil Parameters *(Continued)*

Parameter	Description
`store.quotawarn`	Percentage of quota that is exceeded before clients are warned. Default: 90.
`store.serviceadmingroupdn`	DN of service administrator group.
`store.umask`	umask Default: 077

1. For Messaging Server backward compatibility, not recommended for Sun Java System Messaging Server.

Chapter 4

MTA Configuration

The following topics are covered in this chapter:

The MTA Configuration Files

This section explains the structure and layout of the MTA configuration files. Some configuration modifications are performed by using the command-line interface, as described in Chapter 2, “Message Transfer Agent Command-line Utilities.” Modifications not possible through the command line are performed by editing the configuration files. We recommend that only experienced administrators edit and modify the configuration files.

All configuration files are ASCII text files that are created or changed with any text editor. Permissions for the configuration file should be set to world-readable. Failure to make configuration files world-readable may cause unexpected MTA failures. A physical line in most files is limited to 252 characters and you can split a logical line into multiple physical lines using the backslash (\) continuation character.

NOTE

The MTA processes read most of their configuration from the file .../`config.dat`. This file is the compiled form of the configuration build from the various text configuration files.

Some configuration files are not compiled into this compiled configuration file. In particular, `dispatcher.cnf`, `job_controller.cnf`, and the channel option files, for instance `tcp_local_option`, are not in the compiled configuration, so it is not necessary to compile the configuration to activate changes to these files. However, they are only read by processes when they start. Thus, to activate a change to the job controller's configuration, it is necessary to restart the job controller.

The compiled configuration itself is in two parts. Some, like the rewrite rules and channel definitions, can not be reloaded by running processes. To activate a change to part of the configuration that can not be reloaded, it is necessary to recompile the configuration and then to restart the processes that are affected. For instance, changes to the rewrite rules affect any process that enqueues messages. Thus a change to rewrite rules would require the configuration to be recompiled, and the dispatcher and the job controller to be restarted (thus causing a new generation of tcp_smtp_servers and delivery channel programs to be started.

Some of the configuration, for instance the mappings, aliases, and the general, reverse, and forward lookup tables are reloadable. Changes to these files can be activated by recompiling the configuration and issuing the `imsimta reload` command. The `imsimta reload` command informs all the running processes that they should reload the reloadable part of the compiled configuration.

Table 4-1 lists the MTA configuration files with a short description.

Table 4-1 MTA Configuration files

File	Description
Alias File (mandatory)	Implements aliases not present in the directory. *msg_svr_base*/config/aliases
SMTP Channel Option Files	Sets channel specific options. *msg_svr_base*/config/*channel*_option
Conversion File	Used by conversion channel to control message body part conversions. *msg_svr_base*/config/conversions
Dispatcher Configuration File (mandatory)	Specifies configuration file options for the service dispatcher. *msg_svr_base*/config/dispatcher.cnf
forward.txt (optional)	A text look up file, equivalent in function to the forward database. It provides an alternative mechanism to the LDAP directory for converting to addresses in messages flowing through the system. Setting bit 2 (value 4) of the MTA option USE_TEXT_DATABASES enables the use of this file instead of the reverse database. The file is converted into a hash table that is loaded into memory as part of the reloadable configuration. Only used if the MTA option USE_FORWARD_DATABASE is set
general.txt (optional)	(optional) A general text look up file. This file has the same function as the general database. Setting bit 0 (value 1) of the MTA option USE_TEXT_DATABASES enables use of this file instead of the general database. The file is converted into a hash table that is loaded into memory as part of the reloadable configuration.
Job Controller Configuration File (mandatory)	Defines Job Controller options *msg_svr_base*/config/job_controller.cnf
MTA Configuration File (mandatory)	Defines address rewriting and routing as well as channel definition. *msg_svr_base*/config/imta.cnf
Mapping File (mandatory)	Repository of mapping tables. *msg_svr_base*/config/mappings
Option File	Defines global MTA options. *msg_svr_base*/config/option.dat
reverse.txt (optional)	A text look up file, equivalent in function to the reverse database. It provides an alternative mechanism to the LDAP directory for converting from: addresses in messages flowing through the system. Setting bit 1 (value 2) of the MTA option USE_TEXT_DATABASES enables the use of this file instead of the reverse database. The file is converted into a hash table that is loaded into memory as part of the reloadable configuration. Only used if the MTA option USE_REVERSE_DATABASE is set.
Tailor File (mandatory)	Specifies locations. *msg_svr_base*/config/imta_tailor

Table 4-2 lists the MTA database files with a short description.

Table 4-2 MTA Database Files

File	Description
Reverse Database	Changes from: address in outgoing mail. This provides an alternative mechanism to using the directory, and is for specialized purposes only. An alternative to the reverse database is the reverse lookup table described in Table 4-1.
Forward Database	Changes to: address in outgoing mail. This provides an alternative mechanism to using the directory, and is for specialized purposes only. An alternative to the forward database is the forward lookup table described in Table 4-1.
General Database	Used with domain rewriting rules or in mapping rules, for site-specific purposes. *msg_svr_base*/data/db/generaldb.db. An alternative to the general database is the general lookup table described in Table 4-1.

MTA Configuration File

The MTA configuration file (`imta.cnf`) contains the routing and address rewriting configuration information. It defines all channels and their characteristics, the rules to route mail among those channels, and the method in which addresses are rewritten by the MTA.

Structure of the imta.cnf File

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.

Comments in the File

Comment lines may appear anywhere in the configuration file. A comment is introduced with an exclamation point (!) in column one. Liberal use of comments to explain what is going on is strongly encouraged. The following `imta.cnf` file fragment displays the use of comment lines.

```
! Part I: Rewrite rules
!
ims-ms.my_server.siroe.com $E$U@ims-ms-daemon
!
! Part II: Channel definitions
```

Distinguishing between blank lines and comment lines is important. Blank lines play an important role in delimiting sections of the configuration file. Comment lines are ignored by the configuration file reading routines—they are literally “not there” as far as the routines are concerned and do not count as blank lines.

Including Other Files

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. The following `imta.cnf` file fragment includes the `/usr/iplanet/server5/msg-tango/table/internet.rules` file.

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

NOTE Any files included in the configuration file must be world-readable just as the configuration file is world-readable.

Domain Rewrite Rules

Domain rewrite rules play two important roles:

- Rewrite addresses into their proper form.
- Determine to which channels a message should be enqueued. The determination of which channel to enqueue a message is made by rewriting its envelope To: address.

Each rewrite rule appears on a single line in the upper half of the `imta.cnf` file.

For additional information about configuring rewrite rules, refer to the chapter "Configuring Rewrite Rules" in the *Sun Java System Messaging Server Administration Guide.*

Rewrite Rule Structure

Rewrite rules appear in the upper-half of the MTA configuration file, `imta.cnf`. Each rule in the configuration file appears on a single line. Comments, but not blank lines, are allowed between the rules. The rewrite rules end with a blank line, after which the channel definitions follow. Figure 4-1 shows the rewrite rule section of a partial configuration file.

Figure 4-1 Simple Configuration File - Rewrite Rules

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

! Begin channel definitions
```

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

```
pattern   template
```

pattern

Indicates the string to search for in the domain name. In Figure 4-1, the patterns are a, b, c, and d.

If the pattern matches the domain part of the address, the rewrite rule is applied to the address. A blank space must separate the pattern from the template. For more information about pattern syntax, see “Rewrite Rule Patterns and Tags” on page 171.

template

Is one of the following. For more information about template syntax, see “Rewrite Rule Templates” on page 173.

UserTemplate%DomainTemplate@ChannelTag[*controls*]

UserTemplate@ChannelTag[*controls*]

UserTemplate%DomainTemplate[*controls*]

UserTemplate@DomainTemplate@ChannelTag[*controls*]

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

UserTemplate	Specifies how the user part of the address is rewritten. Substitution sequences can be used to represent parts of the original address or the results of a database lookup. The substitution sequences are replaced with what they represent to construct the rewritten address. In Figure 6-1, the $U substitution sequence is used. For more information, see “Template Substitutions and Rewrite Rule Control Sequences” on page 173.
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. Some can appear almost anywhere in a rule. For more information about controls, see “Template Substitutions and Rewrite Rule Control Sequences” on page 173.

Rewrite Rule Patterns and Tags

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

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

```
host.siroe.com
```

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

```
.siroe.com
```

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

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

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

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

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

This file is required to contain the following:

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

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 4-3, and discussed in the following subsections.

Table 4-3 Summary of Special Patterns for Rewrite Rules

Pattern	Description/Usage
$*	Matches any address. This rule, if specified, is tried first regardless of its position in the file.
$%	Percent Hack Rule. Matches any host/domain specification of the form A%B.
$!	Bang-style Rule. Matches any host/domain specification of the form B!A.
[]	IP literal match-all rule. Matches any IP domain literal.
.	Matches any host/domain specification. For example, `joe@[129.165.12.11]`

In addition to these special patterns, Messaging Server also has the concept of *tags*, which may appear in rewrite rule patterns. These tags are used in situations where an address may be rewritten several times and, based upon previous rewrites, distinctions must be made in subsequent rewrites by controlling which rewrite rules match the address. For more information, see the *Sun Java System Messaging Server Administration Guide*.

Rewrite Rule Templates

Table 4-4 summarizes the template formats.

Table 4-4 Summary of Template Formats for Rewrite Rules

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

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.

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.

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 4-5 summarizes the template substitutions and control sequences.

Table 4-5 Summary of Template Substitutions and Control Sequences

Substitution Sequence	Substitutes
$D	Portion of domain specification that matched.

Table 4-5 Summary of Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
$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.
$[...]	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.

Table 4-5 Summary of Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
$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
$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.

Table 4-5 Summary of Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
$*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`

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

Channel Definitions

The second part of an MTA configuration file contains the definitions for the channels themselves. 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. Each individual channel definition forms a “channel block.” Blocks are separated by single blank lines. Comments (but no blank lines) may appear inside a channel block. A channel block contains a list of keywords which define the configuration of a channel. These keywords are referred to as “channel keywords.” See Table 4-6 for more information.

The following `imta.cnf` file fragment displays a sample channel block:

```
[blank line]
! sample channel block
channelname keyword1 keyword2
routing_system
[blank line]
```

The routing_system is the host name associated with this channel. During the address rewriting process, the host part of the address is checked against the hostnames associated with the channels before any pattern matching in the rewrite rules. The only exception to this is that the $* and exact pattern match rewrite rules are checked first.

For detailed information about channel definitions and channel table keywords, refer to the section "Channel Configuration Keywords" and to Table 4-6.

Channel Configuration Keywords

The first line of each channel block is composed of the channel name, followed by a list of keywords defining the configuration of the specific channel. The following tables describe keywords and how they control various aspects of channel behavior, such as the types of addresses the channel supports. A distinction is made between the addresses used in the transfer layer (the message envelope) and those used in message headers.

The keywords following the channel name are used to assign various attributes to the channel. Keywords are case-insensitive and may be up to 32 characters long; any additional characters are ignored. The supported keywords are listed in Table 4-6 and Table 4-7; the keywords shown in **boldface** are defaults. Table 4-6 lists channel keywords alphabetically; Table 4-7 lists channel keywords by functional group.

Specifying a keyword not on this list is not an error (although it may be incorrect). On UNIX systems, undefined keywords are interpreted as group IDs which are required from a process in order to enqueue mail to the channel. The imsimta test -rewrite utility tells you whether you have keywords in your configuration file that don't match any keywords, and which are interpreted as group ids.

Table 4-6 Channel Keywords Listed Alphabetically

Keyword	Usage
733	Use % routing in the envelope; synonymous with percents. Percent sign envelope addresses. 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. 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. **Syntax**: 733

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
822	Use source routes in the envelope; synonymous with `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. **Syntax**: 822
`addreturnpath`	Adds a Return-path: header when enqueuing to this channel.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. **Syntax**: `addreturnpath` *header* *header* is the header line to be added.
`addrsperfile`	Number of addresses per message file. 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. See `multiple`. **Syntax**: `addrsperfile` *integer* *integer* specifies 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.
`addrsperjob`	Number of addresses to be processed by a single job. The `addrsperjob` keyword computes the number of concurrent jobs to start by dividing the total number of `To:` addressees in all entries by the given value. **Syntax**: `addrsperjob` *integer* *integer* specifies the number of addresses that must be sent to the associated channel before more than one master process is created to handle the addresses. If a value less than or equal to zero is specified, it is interpreted as a request to queue only one service job.
`aliaslocal`	Query 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. Normally only addresses rewritten to the local channel (the l channel on UNIX) are looked up in the alias file and alias database. The exact form of the lookup probes that are performed is then controlled by the `ALIAS_DOMAINS` option. **Syntax**: `aliaslocal`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`aliaspostmaster`	Redirect postmaster messages to the local channel postmaster. If the `aliaspostmaster` keyword is placed on a channel, then any messages addressed to the username `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 has a valid postmaster account that receives mail. So the aliaspostmaster keyword can be useful when it is desired to centralize postmaster responsibilities, rather than setting separate postmaster accounts for separate domains. **Syntax**: `aliaspostmaster`
`allowetrn`	Honor all ETRN commands. This keyword (and associated SMTP ETRN command keywords) control the MTA response when sending a message. The SMTP client issues the SMTP ETRN command, requesting that the MTA attempt to deliver messages in the MTA queues. **Syntax**: `allowetern`
`allowswitchchannel`	Allow the source channel to switch to this channel. **Syntax**: `allowswitchchannel` *channel*
`alternatechannel`	Specify an alternate channel to which to enqueue a message when at least one of `alternateblocklimit`, `alternatelinelimit`, or `alternaterecipientlimit` is exceeded. If any of the `alternate*limit` channel keyword limits is exceeded, the message is diverted to the `alternatechannel`. Using one or more `alternate*limit` keywords without using alternatechannel does not cause an error; instead, it is merely ignored. Therefore, using `alternate*limit` keywords have no effect unless the `alternatechannel` keyword is specified. **Syntax**: `alternatechannel` *channel*
`alternateblocklimit`	Specify the maximum number of MTA blocks allowed per message on the original channel where the `alternatechannel` keyword is placed. Messages exceeding this number of blocks are forced to the channel's `alternatechannel`. Note that the interpretation of block size can be changed in the MTA options file by modifying the `BLOCK_SIZE` option. **Syntax**: `alternateblocklimit` *integer* default: no limit

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`alternatelinelimit`	Specify the maximum number of lines allowed per message on the original channel where the `alternatechannel` keyword is placed. Messages exceeding this number of lines are forced to the channel's `alternatechannel`. **Syntax**: `alternatelinelimit` *integer* default: no limit
`alternaterecipientlimit`	Specify a limit on envelope recipients for a message copy on the original channel where the `alternatechannel` keyword is placed. Messages exceeding this number of envelope recipients on a message copy are forced to the channel's `alternatechannel`. The `alternaterecipientlimit` value is checked before addresses are split up into separate files due to channel keywords such as `addrsperfile`, `single`, or `single_sys`. Consequently, the `alternaterecipientlimit` value is compared against the total number of recipients (of the message in question) being enqueued to the channel in question, rather than being compared against the possibly smaller number of such recipients that may be stored in a particular disk file in the channel in question's queue area. **Syntax**: `alternaterecipientlimit` *integer* default: no limit
`authrewrite`	Use SMTP AUTH information in header. 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. **Syntax**: `authrewrite` *integer* *integer* can be one of the following: 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.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
backoff	Specifies the frequency of message delivery retries of messages unsuccessfully delivered. backoff specifies the interval values between retries of all messages regardless of priority unless overridden by nonurgentbackoff, normalbackoff, or urgentbackoff. **Syntax**: backoff "*interval1*" ["*interval2*"] ["*interval3*"] ["*interval4*"] ["*interval5*"] ["*interval6*"] ["*interval7*"] ["*interval8*"] The *interval* uses ISO 8601P syntax and is as follows: P[*years*Y][*months*M][*weeks*W][*days*D][T[*hours*H][*minutes*M][*seconds*S]] The variables *years*, *months*, *weeks*, *days*, *hours*, *minutes*, and *seconds* are integer values that specify the interval between delivery attempts (the first variable specifies the interval between the initial delivery failure and the first delivery attempt). The alphabetic variable labels (P, Y, M, W, D, H, M, S, and T) are case-insensitive. The initial P is required. The other variables are optional, except that T is required if any time values are specified. Up to eight intervals can be specified with any of the backoff, nonurgentbackoff, normalbackoff, urgentbackoff keywords. The last interval specified is used as the interval for additional retry attempts that may be needed. Deliveries are attempted for a 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. The default intervals between delivery retries attempts in minutes is shown below: urgent: 30, 60, 60, 120, 120, 120, 240 normal: 60, 120, 120, 240, 240, 240, 480 nonurgent: 120, 240, 240, 480, 480, 480, 960 See the *Sun Java System Messaging Server Administration Guide* for complete usage and examples.
bangoverpercent	Group A!B%C as A!(B%C). That is, the bangoverpercent keyword forces "bang" addresses (A!B%C) to interpret A as the routing host and C as the final destination host. 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 FRC 976. **Syntax**: bangoverpercent
bangstyle	Use UUCP! (bang-style) routing in the envelope; synonymous with uucp. 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. **Syntax**: bangstyle

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
bidirectional	Channel is served by both a master and slave program. The bidirectional, master, and slave keywords determines 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. **Syntax**: bidirectional
blocketrn	Do not honor ETRN commands. See allowetrn. **Syntax**: blocketrn
blocklimit	Maximum number of MTA blocks allowed per 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. **Syntax**: blocklimit *integer*
cacheeverything	Cache all connection information and enables all forms of caching. The SMTP channel cache normally records both connection successes and failures. However, this caching strategy is not necessarily appropriate for all situations.The cacheeverything, cachefailures, cachesuccesses, and nocache keywords are provided to adjust the MTA's cache. **Syntax**: cacheeverything
cachefailures	Cache only connection failure information. See cacheeverything. **Syntax**: cachefailures
cachesuccesses	Cache only connection success information. This keyword is equivalent to nocache for channels. See cacheeverything. **Syntax**: cachesuccesses
channelfilter	Specify the location of channel filter file; synonym for destinationfilter. The channelfilter keyword may be used on general MTA channels to specify a channel-level filter to apply to outgoing messages. **Syntax**: channelfilter *filter* The *filter* argument is a required URL that describes the channel filter location.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
charset7	Default character set to associate with 7-bit text messages. 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 (default), ISO-8859-1, ISO-8859-2, and so on. 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. If the appropriate keyword is not specified, no character set name is inserted into the Content-type: header lines. **Syntax**: charset7 *charsetname* The *charsetname* argument specifies the character set name.
charset8	Default character set to associate with 8-bit text messages. 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. See charset7 and charsetesc. **Syntax**: charset8 *charsetname* The *charsetname* argument specifies the character set name.
charsetesc	Default character set to associate with 7-bit text messages containing the escape character. See charset7 and charset8. **Syntax**: charsetesc *charsetname* The charsetname argument specifies the character set name.
checkehlo	Check the SMTP response banner returned by the remote SMTP server for the string “ESMTP.” If this string is found, EHLO is used. If the string is not found, 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. **Syntax**: checkehlo

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
commentinc	Leave comments in message header lines intact. 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 commenttotal keywords. **Syntax**: commentinc
commentmap	Runs comment strings in message header lines through the COMMENT_STRINGS mapping table. See commentinc. **Syntax**: commentmap
commentomit	Remove comments from message header lines. See commentinc. **Syntax**: commentomit
commentstrip	Remove problematic characters from comment fields in message header lines. See commentinc. **Syntax**: commentstrip
commenttotal	Strip comments (material in parentheses) from all header lines, except Received: header lines; this keyword is not normally useful or recommended. See commentinc. **Syntax**: commenttotal
connectalias	Does not rewrite addresses upon message dequeue and deliver to whatever host is listed in the recipient address. **Syntax**: connectalias
connectcanonical	Rewrite addresses upon message dequeue and connect to the host alias for the system to which the MTA would be connected. **Syntax**: connectcanonical

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
copysendpost	Send copies of failures to the postmaster unless the originator address is blank. The postmaster then receives copies of all failed messages except those messages that are actually themselves bounces or notifications. The keywords sendpost, copysendpost, errsendpost, and nosendpost control the sending of failed messages to the postmaster. The default behavior, if none of these keywords is specified, is to send a copy of failed mail messages to the postmaster, unless error returns are completely suppressed with a blank Errors-to: header line or a blank envelope From: address. This default behavior does not correspond to any of the keyword settings. **Syntax**: copysendpost
copywarnpost	Send copies of warnings to the postmaster unless the originator address is blank. In this case, the postmaster receives copies of all warnings of undelivered messages except for undelivered messages that are actually themselves bounces or notifications. The keywords warnpost, copywarnpost, errwarnpost, and nowarnpost are used to control the sending of warning messages to the postmaster. The default behavior, if none of these keywords is specified, is to send a copy of warnings to the postmaster unless warnings are completely suppressed with a blank Warnings-to: header line or a blank envelope From: address. This default behavior does not correspond to any of the keyword settings. **Syntax**: copywarnpost
daemon	Specify the name or IP address of a gateway through which to route mail. The daemon keyword is used on SMTP channels to control the choice of target host. Normally such channels connect to whatever host is listed in the envelop address of the message being processed. The daemon keyword is used to tell the channel to instead connect to a specific remote system, generally a firewall or mailhub system, regardless of the envelope address. **Syntax**: daemon *routing_hostname* or daemon *[IP address]* The actual remote system name should appear directly after the daemon keyword. If the argument after the daemon keyword is not a fully qualified domain name or a domain literal in square brackets, the argument is ignored and the channel connects to the channel's official host.
datefour	Convert date fields in message headers to four-digit years. Two- digit dates with a value less than 50 have 2000 added, while values greater than 50 have 1900 added. **Syntax**: datefour

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`datetwo`	Convert date fields in message headers to two-digit years. The MTA removes 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. **Syntax**: `datetwo`
`dayofweek`	Include day of week in date specifications in date fields in message headers and add this information to date and time headers if it is missing. **Syntax**: `dayofweek`
`defaulthost`	Specify a particular host name to use to complete addresses. This host name is appended to incoming bare user ids. **Syntax**: `defaulthost` *host1* [*host2*] The `defaulthost` keyword must be followed by the domain name (*host1*) to use in completing addresses (in envelope From: addresses and in headers) that come into that channel. An optional second domain name (*host2*) may be specified to use in completing envelope To: addresses. *host2* must include at least one period in its name.
`defaultnameservers`	Use TCP/IP stack's choice of nameservers. **Syntax**: `defaultnameservers`
`defaultmx`	Channel determines whether or not to do MX lookups from network. 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 **Syntax**: `defaultmx`
`deferred`	Honor and implement recognition of deferred delivery dates (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. **Syntax**: `deferred`
`defragment`	Reassemble any MIME-compliant message and partial parts queued to this channel. 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. **Syntax**: `defragment`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
dequeue_removeroute	Removes source routes from envelope To: addresses when dequeuing.The dequeue_removeroute channel keyword can be used on outgoing TCP/IP channels to cause source routes to be removed from envelope recipient addresses. In particular, this keyword may be useful at sites that use the mailhost attribute to direct messages to NMS systems or other systems that do not support source routes. **Syntax**: dequeue_removeroute
destinationbrightmail	Specifies that all messages destined to this channel be subject to Brightmail processing if the recipient has opted in via the LDAP attribute mailAntiUBEService (or equivalent). **Syntax**: destinationbrightmail
destinationbrightmailoptin	Specifies that all messages destined to this channel will be subject to the specified brightmail processing (either spam or virus or both) even if those services have not been opted in by the user or domain via the LDAP attribute. The filter list follows the keyword. The list following must be either spam or virus or spam,virus or virus,spam. Example 1: ims-ms destinationbrightmailoptin spam,virus. . . All mail destined for the message store is scanned for both spam and virus by Brightmail
destinationfilter	Specifies the location of channel filter file that applies to outgoing messages. The destinationfilter is a synonym for channelfilter. **Syntax**: destinationfilter *filter* The *filter* argument is a required URL that describes the channel filter location.
disableetrn	Disable support for the ETRN SMTP command. ETRN is not advertised by the SMTP server as a supported command. See allowetrn. **Syntax**: disableetrn
domainetrn	Tell the MTA to honor only those ETRN commands that specify a domain. The domainetrn keyword also causes the MTA not to echo back the name of the channel that the domain matched and that the MTA be attempts to run. See allowetrn. **Syntax**: domainetrn
domainvrfy	Issue SMTP VRFY command using full address (for example, user@host) as its argument. The domainvrfy, localvrfy, and novrfy keywords control the MTA's use of the VRFY command in its SMTP client. **Syntax**: domainvrfy

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`dropblank`	Strip blank To:, Resent-To, Cc:, or Resent-Cc: headers from incoming messages if specified on a source channel. **Syntax**: `dropblank`
`ehlo`	Use EHLO on all initial SMTP connections. See `checkehlo`. **Syntax**: `ehlo`
`eightbit`	Channel supports 8-bit characters. The `eightbit` keyword should be used on channels that do not restrict the use of characters with ordinal values greater than 127 (decimal). **Syntax**: `eightbit`
`eightnegotiate`	Channel should negotiate use of eight bit transmission, if possible. Some transfers, such as extended SMTP, may actually support a form of negotiation to determine if eight-bit characters can be transmitted. The `eightnegotiate` keyword can be used to instruct the channel to encode messages when negotiation fails. This is the default for all channels; channels that do not support negotiation assume that the transfer is capable of handling eight-bit data **Syntax**: `eightnegotiate`
`eightstrict`	Channel should reject incoming messages with headers that contain illegal eight bit data. **Syntax** `eightstrict`
`errsendpost`	Send copies of failures to the postmaster if the originator address is illegal (cannot be returned). See `copysendpost`. **Syntax**: `errsendpost`
`errwarnpost`	Send copies of warnings to the postmaster if the originator address is illegal (cannot be returned). See `copywarnpost`. **Syntax**: `errwarnpost`
`expandchannel`	Channel in which to perform deferred expansion due to application of `expandlimit`. The reprocessing channel would be used by default, if `expandchannel` were not specified, but use of a processing channel is typically necessary for Messaging Server configurations. If a channel for deferred processing is specified via `expandchannel`, that channel should be a reprocessing or processing channel. However, the Messaging Server typically should be a processing channel; specification of other sorts of channels may lead to unpredictable results. **Syntax**: `expandchannel`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`expandlimit`	Process an incoming message "offline" when the number of addressees exceeds this limit. **Syntax**: `expandlimit` *integer* 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 forces deferred processing on all incoming addresses from the channel.
`expnallow`	Allows `EXPN` even if it has been disabled at the SMTP server level with the `DISABLE_EXPAND` SMTP channel option. **Syntax** `expnallow`
`expndisable`	Disables `EXPN` unconditionally. **Syntax** `expndisable`
`expndefault`	Allows `EXPN` if the SMTP server is set to allow it. **Syntax** `expndefault`
`exproute`	Use explicit routing for this channel's addresses. 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. **Syntax**: `exproute`
`fileinto`	Specify effect on address when a mailbox filter `fileinto` operation is applied. The `fileinto` keyword is currently supported only for ims-ms channels. 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.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`filesperjob`	Number of queue entries to be processed by a single job. The `filesperjob` keyword divides the number of actual queue entries or files by the given value. 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 `filesperjob` and `addrsperjob` keywords can be used to create additional master processes. **Syntax**: `filesperjob` *integer* The argument for `filesperjob` is a single positive integer which specifies the number of addresses or queue entries (files) that must be sent to the associated channel before more than one master process 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 defaults to a value of 0.
`filter`	Specify the location of user filter files. The `filter` keyword may be used on the native and ims-ms channels. **Syntax**: `filter` *url* The argument for filter is a required URL describing the filter file location.
`forwardcheckdelete`	Affects verification of source IP address. The `forwardcheckdelete` keyword tells the MTA to perform 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. Use the original IP address instead. The `fowardchecknone`, `forwardchecktag`, and `forwardcheckdelete` keywords can modify the effects of performing reverse lookups and controlling whether the MTA performs a forward lookup of an IP name found using a DNS reverse lookup. If such forward lookups are requested, these keywords also determine what the MTA does if the forward lookup of the IP name does not match the original IP number of the connection. **Syntax**: `forwardcheckdelete`
`forwardchecknone`	No forward lookup is performed. See `forwardcheckdelete`. **Syntax**: `forwardchecknone`
`forwardchecktag`	Tell the MTA to perform 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. See `forwardcheckdelete`. **Syntax**: `forwardchecktag`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
header_733	Use % routing in the message header. This channel supports RFC 822 format header addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead. 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. **Syntax**: header_733
header_822	Use source routes in the message header. 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. **Syntax**: header_822
header_uucp	Use ! (bang-style) or UUCP routing in the header. The use of this keyword is not recommended. Such usage violates RFC 976. **Syntax**: header_uucp
headerlabelalign	Align header lines for message headers enqueued on this channel. This keyword takes an integer-valued argument. The alignment point is the margin where the contents of headers are aligned. **Syntax**: headerlabelalign *alignment_point* The headerlabelalign keyword takes an integer-valued argument. The alignment point is the margin where the contents of headers are aligned.The default value is 0, which causes headers not to be aligned.
headerlimit	Imposes a limit on the maximum size of the primary (outermost) message header. The primary message headers are silently truncated when the limit is reached. If the global MTA option, HEADER_LIMIT, is set, it overrides this channel-level limit. Default: no limit
headerlinelength	Control the length of message header lines enqueued on this channel. Lines longer than this keyword specifies are folded in accordance with RFC 822 folding rules. **Syntax**: headerlinelength *length* The *length* value is an integer. The default, if this keyword is not explicitly set, is 80. Lines longer than this are folded in accordance with RFC 822 folding rules.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`headerread`	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. **Syntax**: `headerread` *channel*`_read_headers.opt` *channel* is the name of the channel with which the header option file is associated.
`headertrim`	Applies header trimming rules from an options file to the message headers (use with caution) after the original message headers are processed. The `headertrim` keyword impacts only messages that are destined to that channel. Source channels are not impacted. **Syntax**: `headertrim` *channel*`_headers.opt` *channel* is the name of the channel with which the header option file is associated.
`holdlimit`	Mark as .HELD an incoming message when the number of addressees exceeds this limit and enqueue to the reprocess channel (or to whatever channel is specified via the `expandchannel` keyword). As `.HELD` messages, the files sit unprocessed in that MTA queue area awaiting manual intervention by the MTA postmaster. **Syntax**: `holdlimit`
`holdexquota`	Hold messages for users that are over quota. These 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. The `holdexquota` and `noexquota` keywords control the handling of messages addressed to Berkeley mailbox users (UNIX) who have exceeded their disk quota. **Syntax**: `holdexquota`
`identnone`	Disable IDENT lookups; perform IP-to-hostname translation. Both IP number and host name are included in the Received: header lines for the message. **Syntax**: `identnone`
`identnonelimited`	Has the same effect as `identnone` as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header. Where it differs is that with 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. **Syntax**: `identnonelimited`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
identnonenumeric	Disable IDENT lookups and inhibits the usual DNS reverse lookup translation of IP number to host name. This might result in a performance improvement at the cost of less user-friendly information in the Received: header. **Syntax**: identnonenumeric
identnonesymbolic	Disable this IDENT lookup, but does perform IP to host name translation. Only the host name is included in the Received: header for the message. **Syntax**: identnonesymbolic
identtcp	Perform IDENT lookups on incoming SMTP connections and IP to host name translation. The IDENT lookup uses the IDENT protocol (RFC 1413). The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is then inserted into the Received: header lines of the message, with the host name corresponding to the incoming IP number, as reported from a DNS reverse lookup and the IP number itself. **Syntax**: identtcp
identtcplimited	Has the same effect as identtcp as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header. Where it differs from identtcp is that 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 revers lookup succeeds in determining a host name. **Syntax**: identtcplimited
identtcpnumeric	Perform IDENT lookups on incoming SMTP connections; disable IP to hostname translation. **Syntax**: indenttcpnumeric
identtcpsymbolic	Enable IDENT protocol (RFC 1413). The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is then inserted into the Received: header lines of the message, with the actual incoming IP number, as reported from a DNS reverse lookup; the IP number itself is not included in the Received: header. **Syntax**: identtcpsymbolic
ignoreencoding	Ignore Encoding: header on incoming messages. **Syntax**: ignoreencoding
improute	Use implicit routing for this channel's addresses. The improute keyword indicates to the MTA that all addresses matching other channels need routing when they are used in mail sent to a channel marked improute. **Syntax**: improute

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
includefinal	Include final form of address in delivery notifications (recipient address). The includefinal and suppressfinal channel keywords control whether the MTA also includes the final form of the address. **Syntax**: includefinal
inner	Parse messages and rewrite inner message headers. This keyword can be applied to any channel. **Syntax**: inner
innertrim	Apply header trimming rules from an options file to inner message headers for example, embedded MESSAGE/RFC822 headers (use with caution). **Syntax**: innertrim *channel*_headers.opt *channel* is the name of the channel with which the header option file is associated.
interfaceaddress	Bind to the specified TCP/IP interface address as the source address for outbound connections. On a system with multiple interface addresses this keyword controls which address is used as the source IP address when the MTA sends outgoing SMTP messages. Note that it complements the Dispatcher option INTERFACE_ADDRESS, which controls which interface address a TCP/IP channel listens on for accepting incoming connections and messages. **Syntax**: interfaceaddress *address*
interpretencoding	Interpret Encoding: header on incoming messages, if otherwise configured to do so. **Syntax**: interpretencoding
language	Specifies the default language of encoded words in headers. **Syntax**: language *default_language*
lastresort	Specify a host to which to connect 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. **Syntax**: lastresort *host* The keyword requires a single parameter specifying the name of the “system of last resort.”

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`linelength`	Message lines exceeding this length limit are wrapped (MIME encoded). 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 `linelength` keyword causes encoding of data to perform “soft” line wrapping for transport purposes. **Syntax**: `linelength` *length*
`linelimit`	Maximum number of lines allowed per 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. **Syntax**: `linelimit` *integer*
`lmtp`	Specifies that this channel uses LMTP rather than SMTP. Do not use the `smtp` and `lmtp` keywords on the same channel. **Syntax**: `lmtp`
`localvrfy`	Issue SMTP VRFY command using local part of the address. For example, for the address `user1@siroe.com`, `user1` is used with the VRFY command. See `domainvrfy`. **Syntax**: `localvrfy`
`logging`	Log message enqueues and dequeues into the log file and activates logging for a particular channel. Logging is controlled on a per-channel basis. All log entries are made to the file `mail.log_current` in the log directory *msg_svr_base*`/log/imta/mail.log_current`. **Syntax**: `logging`
`loopcheck`	Places a string into the SMTP banner in order for the SMTP server 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 bounces the message if the client is in fact communicating with the SMTP server. **Syntax**: `loopcheck`
`mailfromdnsverify`	Verify that an entry in the DNS exists for the domain used on the SMTP MAIL FROM: command when set on an incoming TCP/IP channel. The MTA rejects the message if no such entry exists. **Syntax**: `mailfromdnsverify`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`master`	Channel is served only by a master program. See `bidirectional`. **Syntax**: `master`
`master_debug`	Generate debugging output in the channel's master program output. Some channel programs include optional code to assist in debugging by producing additional diagnostic output. The `master_debug` and `slave_debug` channel keywords are provided to enable generation of this debugging output on a per-channel basis. On UNIX, when `master_debug` and `slave_debug` is enabled for the l channel, users 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. **Syntax**: `master_debug`
`maxblocks`	Maximum number of MTA blocks per message; longer messages are broken into multiple messages. An MTA block is normally 1024 bytes; this can be changed with the `BLOCK_SIZE` option in the MTA option file. The `maxblocks` and `maxlines` keywords are used to impose size limits beyond which automatic fragmentation are activated. **Syntax**: `maxblocks` *integer*
`maxheaderaddrs`	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines. **Syntax**: `maxheaderaddrs` *integer* This keyword requires 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.
`maxheaderchars`	Maximum number of characters per message header line; longer header lines are broken into multiple header lines. **Syntax**: `maxheaderchars` *integer* This keyword requires 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.
`maxjobs`	Maximum number of concurrent jobs that can be running at one time. If the computed number of service jobs is greater than this value, only `maxjobs` jobs are actually created. Normally `maxjobs` is limited by a value that is less than or equal to the total number of jobs that can run simultaneously in whatever Job Controller pool or pools the channel uses. The default for this value if `maxjobs` is not specified is 100. **Syntax**: `maxjobs` *integer*

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`maxlines`	Maximum number of message lines per message; longer messages are broken into multiple messages. This limit can be imposed simultaneously if necessary. See `maxblocks`. **Syntax**: `maxlines` *integer*
`maxprocchars`	Specifies maximum length of headers to process and rewrite. Messages with headers longer than specified are still accepted and delivered; the only difference is that the long header lines are not rewritten in any way. **Syntax**: `maxprocchars` *integer* The default is processing headers of any length.
`maysaslserver`	Cause the SMTP server to permit clients to attempt to use SASL authentication. The `maysaslserver`, `mustsaslserver`, `nosasl`, `nosaslserver`, `nosaslswitchchannel`, and `saslswitchchannel` keywords are used to configure SASL (SMTP AUTH) use during the SMTP protocol by SMTP channels such as TCP/IP channels. **Syntax**: `maysaslserver`
`maytls`	SMTP client and server allow TLS use to incoming connections and to attempt TLS upon outgoing connections. 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. **Syntax**: `maytls`
`maytlsclient`	SMTP client attempts TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS. See `maytls`. **Syntax**: `maytlsclient`
`maytlsserver`	SMTP server allows TLS use and advertises support for the STARTTLS extension when receiving messages. See `maytls`. **Syntax**: `maytlsserver`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`missingrecipientpolicy`	Controls handling of messages missing recipient header lines. **Syntax**: `missingrecipientpolicy` *integer* 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. The values for `missingrecipientpolicy` are: • 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.
`msexchange`	Serves channel for Microsoft Exchange gateways and clients. The `msexchange` channel keyword also causes advertisement (and recognition) of broken TLS commands. **Syntax**: `msexchange`
`multiple`	Accept multiple destination hosts in a single message copy for the entire channel. Note that at least one copy of each message is created for each channel the message is queued to, regardless of the keywords used. The 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. The keywords `multiple`, `addrsperfile`, `single`, and `single_sys` can be used to control how multiple addresses are handled. **Syntax**: `multiple`
`mustsaslserver`	Cause the SMTP server to insist that clients use SASL authentication; the SMTP server does not accept messages unless the remote client successfully authenticates. See `maysaslserver`. **Syntax**: `mustsaslserver`
`musttls`	SMTP client and server insist upon TLS use n both outgoing and incoming connections and does not transfer messages with remote sides that do not support TLS. Email is not exchanged with remote systems that fail to successfully negotiate TLS use. See `maytls`. **Syntax**: `musttls`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
musttlsclient	SMTP client insists upon TLS use when sending outgoing messages and does not send messages to any remote SMTP server that does not support TLS use. See maytls. **Syntax**: musttlsclient
musttlsserver	SMTP server insists upon TLS use and does not accept messages from any remote SMTP client that does not support TLS use. See maytls. **Syntax**: musttlsserver
mx	TCP/IP network and software supports MX record lookups. The mx keyword is currently equivalent to nonrandommx. See randommx. **Syntax**: mx
nameparameterlengthlimit	Controls the points at which the name content-type and filename content-disposition parameters are truncated. See parameterlengthlimit. Default: 128 **Syntax**: nameparameterlengthlimit *integer*
nameservers	Consult specified nameservers rather than TCP/IP stack's choice when nameserver lookups are being performed, that is, unless the nsswitch.conf file on UNIX or the Windows NT TCP/IP configuration selects no use of nameservers. **Syntax**: nameservers *IP_address1 IP_address2 ...* nameservers requires a space separated list of IP addresses for the nameservers.
noaddreturnpath	Do not add a Return-path: header when enqueuing to this channel.
nobangoverpercent	Group A!B%C as (A!B)%C (default). That is, the nobangoverpercent keyword forces "bang" addresses (A!B%C) to interpret C as the routing host and A as the final destination host. 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 FRC 976. **Syntax**: nobangoverpercent
noblocklimit	No limit specified for the number of MTA blocks allowed per message. See blocklimit. **Syntax**: noblocklimit
nocache	Do not cache any connection information. See cacheeverything. **Syntax**: nocache

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
nochannelfilter	Do not perform channel filtering for outgoing messages; synonym for nodestinationfilter. See channelfilter. **Syntax**: nochannelfilter
nodayofweek	Remove day of week from date/time specifications. 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. See dayofweek. **Syntax**: nodayofweek
nodefaulthost	Do not specify a domain name to use to complete addresses. See defaulthost. **Syntax**: nodefaulthost
nodeferred	Do not honor deferred delivery dates. See deferred. **Syntax**: nodeferred
nodefragment	Do not perform special processing for message/partial messages. See defragment. **Syntax**: nodefragment
nodestinationfilter	Do not perform channel filtering for outgoing messages. See destinationfilter. **Syntax**: nodestinationfilter
nodropblank	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers. See dropblank. **Syntax**: nodropblank
noehlo	Never use the SMTP EHLO command. See ehlo. **Syntax**: noehlo
noexproute	No explicit routing for this channel's addresses. See exproute. **Syntax**: noexproute
noexquota	Return to originator any messages to users who are over quota. See holdexquota. **Syntax**: noexquota
nofileinto	Mailbox filter fileinto operator has no effect. See fileinto. **Syntax**: nofileinto

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
nofilter	Do not perform user mailbox filtering. See `filter`. **Syntax**: `nofilter`
noheaderread	Do not apply header trimming rules from option file upon message enqueue. See `headerread`. **Syntax**: `noheaderread`
noheadertrim	Do not apply header trimming rules from options file. See `headertrim`. **Syntax**: `noheadertrim`
noimproute	No implicit routing for this channel's addresses. See `improute`. **Syntax**: `noimproute`
noinner	Do not rewrite inner message headers. See `inner`. **Syntax**: `noinner`
noinnertrim	Do not apply header trimming to inner message headers. See `innertrim`. **Syntax**: `noinnertrim`
nolinelimit	No limit specified for the number of lines allowed per message. See `linelimit`. **Syntax**: `nolinelimit`
nologging	Do not log message enqueues and dequeues into the log file. See `logging` **Syntax**: `nologging`
noloopcheck	Instructs the SMTP client not check the value of any XLOOP parameter in the EHLO server response to see if the SMTP client is communicating with the SMTP server on the same machine. **Syntax**: `noloopcheck`
nomailfromdnsverify	The MTA does not verify that an entry in the DNS exists for the domain used. See `mailfromdnsverify`. **Syntax**: `nomailfromdnsverify`
nomaster_debug	Do not generate debugging output in the channel's master program output. See `master_debug`. **Syntax**: `nomaster_debug`
nomsexchange	Channel does not serve MS Exchange gateways. See `msexchange`. **Syntax**: `nomsexchange`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
nomx	TCP/IP network does not support MX lookups. See mx. **Syntax**: nomx
nonrandommx	Perform MX lookups; does not randomize returned entries of equal precedence—they should be processed in the same order in which they are received. Equivalent to mx. See also randommx. **Syntax**: nonrandommx
nonurgentbackoff	Specifies the frequency for attempted delivery of nonurgent messages. See backoff. **Syntax**: nonurgentbackoff "*interval1*" ["*interval2*"] ["*interval3*"] ["*interval4*"] ["*interval5*"] ["*interval6*"] ["*interval7*"] ["*interval8*"] The *interval* uses ISO 8601P syntax and is as follows: P[*years*Y][*months*M][*weeks*W][*days*D][T[*hours*H][*minutes*M][*seconds*S]] The variables *years*, *months*, *weeks*, *days*, *hours*, *minutes*, and *seconds* are integer values that specify the interval between delivery attempts (the first variable specifies the interval between the initial delivery failure and the first delivery attempt). The alphabetic variable labels (P, Y, M, W, D, H, M, S, and T) are case-insensitive. The initial P is required. The other variables are optional, except that T is required if any time values are specified. See backoff.
nonurgentblocklimit	Force messages above the specified size to wait unconditionally for a periodic job. The nonurgentblocklimit keyword instructs the MTA to downgrade messages larger than the specified size to lower than nonurgent priority (second class priority). **Syntax**: nonurgentblocklimit *integer*
nonurgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of non-urgent priority. Different return handling for messages of different priorities may be explicitly set using the nonurgentnotices, normalnotices, or urgentnotices keywords. Otherwise, the notices keyword values are used for all messages. See notices. **Syntax**: nonurgentnotices *age1* [*age2*] [*age3*] [*age4*] [*age5*] The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the RETURN_UNITS option is 0 or not specified in the option file; or hours if the RETURN_UNITS option is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (bounced).

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
noreceivedfor	Do not include Envelope to address in Received: header line. The noreceivedfor keyword instructs the MTA to construct Received: header lines without including any envelope addressee information. See receivedfor. **Syntax**: noreceivedfor
noreceivedfrom	Construct Received: header lines without including the original envelope From: address. The noreceivedfrom keyword instructs the MTA to construct Received: header lines without including the original envelope From: address. See receivedfrom. **Syntax**: noreceivedfrom
noremotehost	Use local host's domain name as the default domain name to complete addresses. See remotehost. **Syntax**: noremotehost
norestricted	Do not apply RFC 1137 restricted encoding to addresses. Equivalent to unrestricted keyword. See restricted. **Syntax**: norestricted
noreturnaddress	Use the RETURN_ADDRESS option value. See returnaddress. **Syntax**: noreturnaddress
noreturnpersonal	Use the RETURN_PERSONAL option value. See returnpersonal. **Syntax**: noreturnpersonal
noreverse	Do not apply reverse database to addresses. noreverse exempts addresses in messages queued to the channel from address reversal processing. See reverse. **Syntax**: noreverse

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage

`normalbackoff`	Specifies the frequency for attempted delivery of normal messages. See `backoff`.
	
	Syntax:
	`normalbackoff "`*interval1*`" ["`*interval2*`"] ["`*interval3*`"] ["`*interval4*`"] ["`*interval5*`"] ["`*interval6*`"] ["`*interval7*`"] ["`*interval8*`"]`
	
	The *interval* uses ISO 8601P syntax and is as follows:
	
	`P[`*years*`Y][`*months*`M][`*weeks*`W][`*days*`D][T[`*hours*`H][`*minutes*`M][`*seconds*`S]]`
	
	The variables *years*, *months*, *weeks*, *days*, *hours*, *minutes*, and *seconds* are integer values that specify the interval between delivery attempts (the first variable specifies the interval between the initial delivery failure and the first delivery attempt). The alphabetic variable labels (`P`, `Y`, `M`, `W`, `D`, `H`, `M`, `S`, and `T`) are case-insensitive. The initial `P` is required. The other variables are optional, except that `T` is required if any time values are specified.
	
	See `backoff`.

`normalblocklimit`	Downgrade messages larger than the specified size to nonurgent priority. **Syntax**: `normalblocklimit` *integer*
`normalnotices`	Specify the amount of time which may elapse before notices are sent and messages returned for messages of normal priority. See `notices`. **Syntax**: `normalnotices` *age1* [*age2*] [*age3*] [*age4*] [*age5*] The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the `RETURN_UNITS` option is 0 or not specified in the option file; or hours if the `RETURN_UNITS` option is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (bounced).
`norules`	Do not perform channel-specific rewrite rule checks. This keyword is usually used for debugging and is rarely used in actual applications. See `rules`. **Syntax**: `norules`
`nosasl`	SASL authentication is not permitted or attempted. Do not allow switching to this channel upon successful SASL authentication. See `maysaslserver`. **Syntax**: `nosasl`
`nosaslserver`	SASL authentication is not permitted. See `maysaslserver`. **Syntax**: `nosaslserver`
`nosendetrn`	Do not send an ETRN command. See `sendetrn`. **Syntax**: `nosendetrn`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
nosendpost	Do not send copies of failures to the postmaster. See sendpost. **Syntax**: nosendpost
noservice	Service conversions for messages coming into this channel must be enabled via CHARSET_CONVERSIONS. See service. **Syntax**: noservice
noslave_debug	Do not generate slave debugging output. See slave_debug. **Syntax**: noslave_debug
nosmtp	Channel does not use SMTP. See smtp. **Syntax**: nosmtp
nosourcefilter	Do not perform channel filtering for incoming messages. See sourcefilter. **Syntax**: nosourcefilter
noswitchchannel	Do not switch to the channel associated with the originating host; does not permit being switched to. See switchchannel. **Syntax**: noswitchchannel
notices	Specifies the amount of time that may elapse before notices are sent and messages returned. **Syntax**: notices *age1* [*age2*] [*age3*] [*age4*] [*age5*] The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the RETURN_UNITS option is 0 or not specified in the option file; or hours if the RETURN_UNITS option is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (bounced). When a message attains any of the other ages, a warning notice is sent. The default if no keyword is given is to use the notices setting for the local channel. If no setting has been made for the local channel, then the defaults 3, 6, 9, 12 are used, meaning that warning messages are sent when the message attains the ages 3, 6, and 9 days (or hours) and the message is returned after remaining in the channel queue for more than 12 days (or hours).
notls	SMTP client and server neither attempt nor allow TLS use. See maytls. **Syntax**: notls

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
notlsclient	SMTP client does not attempt TLS use when sending messages. See maytlsclient. **Syntax**: notlsclient
notlsserver	SMTP server does not offer or allow TLS use when receiving messages. See maytlsserver. **Syntax**: notlsserver
novrfy	Do not issue SMTP VRFY commands. See vrfyallow. **Syntax**: novrfy
nowarnpost	Do not send copies of warnings to the postmaster. See warnpost. **Syntax**: nowarnpost
nox_env_to	Do not add X-Envelope-to header lines while enqueuing. See x_env_to. **Syntax**: nox_env_to
parameterlengthlimit	Controls the points at which general content-type and content-disposition parameters are truncated. See nameparameterlengthlimit. Default: 1024 **Syntax**: parameterlengthlimit *integer*
percentonly	Ignores bang paths in address of the form A!B%C. When this keyword is set, percents are interpreted for routing. **Syntax**: percentonly
percents	Use % routing in the envelope; synonymous with 733. **Syntax**: percents
personalinc	Leave personal name fields in message header lines intact when rewriting addresses. 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. **Syntax**: personalinc

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`personalmap`	Run personal names through `PERSONAL_NAMES` mapping table. See `personalinc`. **Syntax**: `personalmap`
`personalomit`	Remove personal name fields from message header lines. See `personalinc`. **Syntax**: `personalomit`
`personalstrip`	Strip problematic characters from personal name fields in message header lines. See `personalinc`. **Syntax**: `personalstrip`
`pool`	Specifies processing pool master channel in which programs run. The MTA creates service jobs (channel master programs) to deliver messages. The Job Controller, which launches these jobs, associates them with pools. Pool types are defined in the `job_controller.cnf` file. The pool with which each channel's master program is associated can be selected on a channel-by-channel basis, using the `pool` keyword. **Syntax**: `pool` *pool_name* The `pool` keyword must be followed by the name of the pool to which delivery jobs for the current channel should be queued. The name of the pool should not contain more than 12 characters. If the `pool` keyword is omitted, then the pool used is the default pool, the first queue listed in the Job Controller configuration file.
`port`	Connect to the specified TCP/IP port. 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. **Syntax**: `port` *port_number*
`postheadbody`	Both the message's header and body are sent to the postmaster when a delivery failure occurs. **Syntax**: `postheadbody`
`postheadonly`	Only the message's header is sent to the postmaster when a delivery failure occurs. **Syntax**: `postheadonly`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
randommx	Perform MX lookups. MX record values of equal precedence should be processed in random order. 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 to which the MTA system is connected. The MTA randomizes the order of returned MX records of equal preference regardless of the mx/randommx/nonrandommx setting **Syntax**: randommx
receivedfor	Includes envelope To: address in Received: head if a message is addressed to just one envelope recipient. **Syntax**: receivedfor
receivedfrom	Include the original envelope From: address when constructing Received: header lines if the MTA has changed the envelope From: address due to, for example, certain sorts of mailing list expansions. **Syntax**: receivedfrom
rejectsmtp	Rejects messages that contain lines longer than 1000 characters (including CRLF). Reject the line when it is over 1000 characters. If the rejectsmtp keyword is placed on a channel, a line over 1000 characters (including CRLF) is rejected. This keyword must be applied to the initial channel used for submission (such as tcp_local). It will not affect any channel that is switched to subsequently. See truncatesmtp and wrapsmtp. **Syntax**: rejectsmtp
remotehost	Use remote host's name as the default domain name to complete addresses. The use of the remote host's domain name is appropriate when dealing with improperly configured SMTP clients. **Syntax**: remotehost
restricted	Apply RFC 1137 restricted encoding to addresses. 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 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. **Syntax**: restricted

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`returnaddress`	Set the return address for the local Postmaster. By default, the Postmaster's return address that is used when the MTA constructs bounce or notification messages is `postmaster@`*local-host*, where *local-host* is the official local host name (the name on the local channel). **Syntax**: `returnaddress` *postmaster_address* `returnaddress` takes a required argument specifying the Postmaster address.
`returnenvelope`	Control use of blank envelope return addresses. **Syntax**: `returnenvelope` *bit_flag* 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.
`returnpersonal`	Set the personal name for the local Postmaster. By default, the Postmaster's personal name that is used when the MTA constructs bounce or notification messages is "MTA e-Mail Interconnect." **Syntax**: `returnpersonal` *postmaster_name* `returnpersonal` takes a required argument specifying the Postmaster personal name.
reverse	Apply reverse database or REVERSE mapping to addresses in messages queued to the channel. **Syntax**: `reverse`
`routelocal`	Attempt short-circuit routing to any explicit routing in addresses when rewriting an address to the channel. 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 % our other routing. **Syntax**: `routelocal`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
rules	Perform channel-specific rewrite rule checks. Usually used for debugging. **Syntax**: rules
saslswitchchannel	Cause incoming connections to be switched to a specified channel upon a client's successful use of SASL. **Syntax**: saslswitchchannel *channel* The *channel* argument specifies the channel to which to switch.
sendpost	Sends copies of failed messages to the postmaster. See copysendpost. **Syntax**: sendpost
sendetrn	Send an ETRN command, if the remote SMTP server says it supports ETRN. The sendetrn and nosendetrn keywords control whether the MTA SMTP client sends an ETRN command at the beginning of an SMTP connection or does not sent an ETRN command at all. **Syntax**: sendetrn *host* The sendetrn keyword should be followed by the name of the system requesting that its messages receive a delivery attempt.
sensitivitycompanyconfidential	Allow messages of any sensitivity. The sensitivity keywords set an upper limit on the sensitivity of messages that can be accepted by a channel. 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 reject when enqueued to the channel with an error message. Note that the MTA performs this sort of sensitivity checking at a per-message, not per-recipient, level. If a desalination 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. **Syntax**: sensitivitycompanyconfidential
sensitivitynormal	Reject messages whose sensitivity is higher than normal. See sensitivitycompanyconfidential. **Syntax**: sensitivitynormal
sensitivitypersonal	Reject messages whose sensitivity is higher than personal. See sensitivitycompanyconfidential. **Syntax**: sensitivitypersonal
sensitivityprivate	Reject messages whose sensitivity is higher than private. See sensitivitycompanyconfidential. **Syntax**: sensitivityprivate.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
service	Perform service conversions for messages coming into the channel. The service keyword unconditionally enables service conversions regardless of CHARSET-CONVERSION entry. **Syntax**: service
sevenbit	Channel does not support 8-bit characters; 8-bit characters must be encoded. The MTA 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 on a given channel by specifying the sevenbit keyword. **Syntax**: sevenbit
silentetrn	Honor all ETRN commands, but without echoing the name of the channel that the domain matched and that the MTA attempts to run. See allowetrn. **Syntax**: silentetrn
single	Only one envelope To: address per message copy or destination address on the channel. See multiple. **Syntax**: multiple
single_sys	Each message copy must be for a single destination system. See multiple. **Syntax**: single_sys
slave	Channel is serviced only by a slave program. See bidirectional. **Syntax**: slave
slave_debug	Generate debugging output in slave programs. See master_debug. **Syntax**: slave_debug
smtp	Channel uses SMTP. The smtp keywords specify whether or not a channel supports the SMTP protocol and what type of SMTP line terminator the MTA expects to see as part of that protocol. The smtp keyword or one of the other smtp_* keywords is mandatory for all SMTP channels. The keywords smtp_cr, smtp_crlf, smtp_crorlf, and smtp_lf can be used on SMTP channels to not only select use of the SMTP protocol, but also to further specify the character sequences to accept as line terminators. It is normal to use CRLF sequences as the SMTP line terminator, and this is what the MTA always generates; these keywords only affect the handling of incoming material. The smtp keyword is synonymous to the smtp_crlf keyword. **Syntax**: smtp

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`smtp_cr`	Accept CR as an SMTP line terminator. See `smtp`. **Syntax**: `smtp_cr`
`smtp_crlf`	Require CRLF as the SMTP line terminator. This means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. See `smtp`. **Syntax**: `smtp_crlf`
`smtp_crorlf`	Allow any of CR (carriage return), LF (line feed), or full CRLF as the SMTP line terminator. See `smtp`. **Syntax**: `smtp_crorlf`
`smtp_lf`	Accept LF (linefeed) without a preceding CR (carriage return) as an SMTP line terminator. See `smtp`. **Syntax**: `smtp_lf`
`sourceblocklimit`	Maximum number of MTA blocks allowed per incoming message. The MTA rejects attempts to submit a message containing more blocks than this to the channel. See `blocklimit`. **Syntax**: `sourceblocklimit` *integer*
`sourcebrightmail`	Specifies that all messages originating from this channel receive Brightmail processing. All recipient addresses will be made known to Brightmail regardless of destination channel if the recipient or the recipient's domain has opted in via the LDAP attribute. Looks at recipient's LDAP attribute `mailAntiUBEService` (or equivalent) to determine whether spam, virus or both or none are filtered. If `mailAntiUBEService` doesn't specify either spam or virus, then mail is not sent to the the Brightmail server for filtering. This should be placed on the switched-to channel, if `switchchannel` is in effect. **Syntax**: `sourcebrightmail`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
sourcebrightmailoptin	Specifies that all messages originating from this channel will be subject to the specified brightmail processing (either spam or virus or both) even if those services have not been opted in by the user or domain via the LDAP attribute.The system-wide default filter list follows the keyword. The list following must be either spam or virus or spam,virus or virus,spam. This should be placed on the switched-to channel, if switchchannel is in effect. Example 1: tcp_local sourcebrightmailoptin spam,virus . . . Specifies that mail be scanned for both spam and virus by Brightmail regardless of the user's LDAP attribute. Example 2: tcp_local sourcebrightmailoptin virus . . . Specifies that mail will default to only virus scanning. In this case, spam filtering can be enabled on a per user basis, or by destination domain via the LDAP attributes.
sourcecommentinc	Leave comments in incoming message header lines. 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. On source channels, this behavior is controlled by the use of the sourcecommentinc, sourcecommentmap, sourcecommentomit, sourcecommentstrip, and sourcecommenttotal keywords. **Syntax**: sourcecommentinc
sourcecommentmap	Runs comment strings in message header lines through source channels. See sourcecommentinc. **Syntax**: sourcecommentmap
sourcecommentomit	Remove comments from incoming message header lines, for example, To:, From:, and Cc: headers. See sourcecommentinc. **Syntax**: sourcecommentomit
sourcecommentstrip	Remove problematic characters from comment field in incoming message header lines. See sourcecommentinc. **Syntax**: sourcecommentstrip
sourcecommenttotal	Strip comments (material in parentheses) everywhere in incoming messages. The sourcecommenttotal keyword indicates to the MTA to remove any comments from all headers, except Received: headers. This keyword is not normally useful or recommended. See sourcecommentinc. **Syntax**: sourcecommenttotal

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`sourcefilter`	Specify the location of channel filter file for incoming messages. **Syntax**: `sourcefilter` *filter* The *filter* argument is a required URL that describes the channel filter location.
`sourcepersonalinc`	Leave personal names in incoming message header lines intact. 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. On source channels, this behavior is controlled by the use of the `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, and `sourcepersonalstrip` keywords. **Syntax**: `sourcepersonalinc`
`sourcepersonalmap`	Run personal names through source channels. See `sourcepersonalinc`. **Syntax**: `sourcepersonalmap`
`sourcepersonalomit`	Remove personal name fields from incoming message header lines. See `sourcepersonalinc`. **Syntax**: `sourcepersonalomit`
`sourcepersonalstrip`	Strip problematic characters from personal name fields in incoming message header lines. See `sourcepersonalinc`. **Syntax**: `sourcepersonalstrip`
`sourceroute`	Use source routes in the message envelope; synonymous with 822. **Syntax**: `sourceroute`
`streaming`	Specify degree of protocol streaming for channel to use. **Syntax**: `streaming 0|1|2|3` This keyword requires an integer parameter; how the parameter is interpreted is specific to the protocol in use. 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.

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`subaddressexact`	Alias must match exactly, including exact subaddress match. 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 wildcarded comparisons or comparisons with the subaddress removed) are performed. **Syntax**: `subaddressexact`
`subaddressrelaxed`	Alias without subaddress may match. 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. The `subaddressrelaxed` keyword is the default. **Syntax**: `subaddressrelaxed`
`subaddresswild`	Alias with subaddress wildcard may match. 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+*. **Syntax**: `subaddresswild`
`subdirs`	Use multiple subdirectories. **Syntax**: `subdirs` *integer* The keyword should be followed by an integer that specifies the number of subdirectories across which to spread messages for the channel.
`submit`	Marks the channel as a submit-only channel. This is normally useful 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 message submissions. **Syntax**: `submit`
`suppressfinal`	Suppress the final address form from notification messages, if an original address form is present, from notification messages. See `includefinal`. **Syntax**: `suppressfinal`
`switchchannel`	Switch from the server channel to the channel associated with the originating host. If `switchchannel` is specified on the initial channel the server uses, the IP address of the connecting (originating) host is matched against the channel table; if it matches, the source channel changes 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 performing a DNS reverse lookup. **Syntax**: `switchchannel`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`threaddepth`	Number of messages per thread. The `threaddepth` keyword may be used to instruct the MTA's 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 normally all handled in one thread). Default: 10 **Syntax**: `threaddepth` *integer*
`tlsswitchchannel`	Switch to specified channel upon successful TLS negotiation. See `maytls`. **Syntax**: `tlsswitchchannel` *channel* The *channel* parameter specifies the channel to which to switch.
`transactionlimit`	Provides functionality equivalent to the `ALLOW_TRANSACTIONS_PER_SESSION` SMTP channel option (See Table 4-9) on a per-channel basis. The default is no limit. **Syntax** `transactionlimit` *integer*
`truncatesmtp`	Truncate the line when it is over 1000 characters. If the `truncatesmtp` keyword is placed on a channel, a line over 1000 characters is truncated. This keyword must be applied to the initial channel used for submission (such as tcp_local). It will not affect any channel that is switched to subsequently. See `rejectsmtp` and `wrapsmtp`. **Syntax**: `truncatesmtp`
`unrestricted`	Do not apply RFC 1137 restricted encoding to addresses. See `restricted`. **Syntax**: `unrestricted`

`urgentbackoff`	Specify the frequency for attempted delivery of urgent messages. See `backoff`.
	Syntax:
	`urgentbackoff "`*interval1*`" ["`*interval2*`"] ["`*interval3*`"] ["`*interval4*`"] ["`*interval5*`"] ["`*interval6*`"] ["`*interval7*`"] ["`*interval8*`"]`
	The *interval* uses ISO 8601P syntax and is as follows:
	`P[`*years*`Y][`*months*`M][`*weeks*`W][`*days*`D][T[`*hours*`H][`*minutes*`M][`*seconds*`S]]`
	The variables *years*, *months*, *weeks*, *days*, *hours*, *minutes*, and *seconds* are integer values that specify the interval between delivery attempts (the first variable specifies the interval between the initial delivery failure and the first delivery attempt). The alphabetic variable labels (`P`, `Y`, `M`, `W`, `D`, `H`, `M`, `S`, and `T`) are case-insensitive. The initial `P` is required. The other variables are optional, except that `T` is required if any time values are specified.

`urgentblocklimit`	Force messages larger the specified size to normal priority. **Syntax**: `urgentblocklimit`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`urgentnotices`	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority. See `notices`. **Syntax**: `urgentnotices` *age1* [*age2*] [*age3*] [*age4*] [*age5*] The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the `RETURN_UNITS` option is 0 or not specified in the option file; or hours if the `RETURN_UNITS` option is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (bounced).
`useintermediate`	Present the address as originally presented to the MTA for notification messages. **Syntax**: `useintermediate`
`user`	Specify the queue for master channel program processing of urgent messages. The `user` keyword is used on pipe channels to indicate under what username to run. **Syntax**: `user` *username* Note that the argument to `user` is normally forced to lowercase, but original case is preserved if the argument is quoted.
`uucp`	Use UUCP! (bang-style) routing in the envelope; synonymous with `bangstyle`. **Syntax**: `uucp`
`viaaliasoptional`	Specify that final recipient addresses that match the channel are not required to be produced by an alias. **Syntax**: `viaaliasoptional`
`viaaliasrequired`	Specify 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 sufficient 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). **Syntax**: `viaaliasrequired`

Table 4-6 Channel Keywords Listed Alphabetically *(Continued)*

Keyword	Usage
`vrfyallow`	Issue a detailed, informative response for SMTP VRFY command. The `vrfyallow`, `vrfydefault`, and `vrfyhide` keywords control the MTA SMTP server's response when a sending SMTP client issues a SMTP VRFY command. 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. **Syntax**: `vrfyallow`
`vrfydefault`	Provide a detailed, informative response for SMTP VRFY command, unless the channel option HIDE_VERIFY=1 has been specified. See `vrfyallow`. **Syntax**: `vrfydefault`
`vrfyhide`	Issue only a vague, ambiguous response to SMTP VRFY command. See `vrfyallow`. **Syntax**: `vrfyhide`
`warnpost`	Send copies of warnings to the postmaster. See `copywarnpost`. **Syntax**: `warnpost`
`wrapsmtp`	Wrap the line instead of truncating it. If the `wrapsmtp` keyword is placed on a channel, a long line (over 1000 characters) wraps to the next line. This keyword must be applied to the initial channel used for submission (such as tcp_local). It will not affect any channel the is switched to subsequently. See `rejectsmtp` and `truncatesmtp`. **Syntax**: `wrapsmtp`
`x_env_to`	Add X-Envelope-to header lines while enqueuing. 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. **Syntax**: `x_env_to single` The `x_env_to` keyword requires the `single` keyword in order to take effect.

Table 4-7 lists channel keywords for functional group.

For additional description about the channel keyword functionality groups, see the chapter "Configuring Channel Definitions" in the *Sun Java System Messaging Server Administration Guide*.

Table 4-7 Channel Keywords Grouped by Functionality

Functionality	Associated Keywords
Address types	733, 822, uucp, header_733, header_822, header_uucp
Address interpretation	bangoverpercent, nobangoverpercent, percentonly
Alternate channels	alternatechannel, alternateblocklimit, alternatelinelimit, alternaterecipientlimit
Brightmail	destinationbrightmail, destinationbrightmailoptin, sourcebrightmail, sourcebrightmailoptin
Routing information in addresses	exproute, improute, noexproute, noimproute
Short circuiting rewriting of routing addresses	routelocal
Address rewriting upon message dequeue	connectalias, connectcanonical
Channel-specific rewrite rules	norules, rules
Channel directionality	bidirectional, master, slave
Message size affection priority	nonurgentblocklimit, normalblocklimit, urgentblocklimit
Channel connection information caching	cacheeverything, cachefailures, cachesuccesses, nocache
Address and message file processing amounts	addrsperjob, filesperjob, maxjobs
Multiple addresses	addrsperfile, multiple, single, single_sys
Expansion of multiple addresses	expandchannel, expandlimit, holdlimit
Multiple subdirectories	subdirs
Service job queue scheduling	pool, maxjobs
Deferred delivery dates	deferred, nodeferred
Undeliverable message notification times	nonurgentnotices, normalnotices, notices, urgentnotices
Returned messages	copysendpost, errsendpost, nosendpost, sendpost
Warning messages	copywarnpost, errwarnpost, nowarnpost, warnpost

Table 4-7 Channel Keywords Grouped by Functionality *(Continued)*

Functionality	Associated Keywords
Postmaster returned message content	postheadbody, postheadonly
Including altered addresses in notification messages	includefinal, suppressfinal, useintermediate
Protocol streaming	streaming
Triggering new threads in multithreaded channels	threaddepth
Channel protocol selection	nosmtp, smtp, smtp_cr, smtp_crlf, smtp_crorlf, smtp_lf
SMTP EHLO command	checkehlo, ehlo, noehlo
Receiving an SMTP ETRN command	allowetrn, blocketrn, disableetrn, domainetrn, silentetrn
Sending an SMTP ETRN command	nosendetrn, sendetrn
SMTP VRFY commands	domainvrfy, localvrfy, novrfy
Responding to SMTP VRFY commands	vrfyallow, vrfydefault, vrfyhide
SMTP EXPN commands	expnallow, expndisable, expndefault
TCP/IP port number	interfaceaddress, port
TCP/IP MX record support	defaultmx, defaultnameservers, mx, nameservers, nomx, nonrandommx, randommx
Last resort host specification	lastresort
Reverse DNS and IDENT lookups on incoming SMTP connections	forwardcheckdelete, forwardchecknone, forwardchecktag, identnone, identnonelimited, identnonenumeric, identnonesymbolic, identtcp, identtcplimited, identtcpnumeric, identtcpsymbolic
Alternate channels for incoming mail	allowswitchchannel, noswitchchannel, switchchannel
Host name for incomplete addresses	defaulthost, nodefaulthost, noremotehost, remotehost
Illegal blank recipient headers	dropblank, nodropblank
Messages without recipient header	missingrecipientpolicy
Eight-bit capability	eightbit, eightnegotiate, eightstrict, sevenbit
Character set labeling	charset7, charset8, charsetesc
Message line length restrictions	linelength

Table 4-7 Channel Keywords Grouped by Functionality *(Continued)*

Functionality	Associated Keywords
Channel-specific use of the reverse database	noreverse, reverse
Inner header rewriting	inner, noinner
Restricted mailbox encoding	norestricted, restricted, unrestricted
Message header line trimming	headerread, headertrim, innertrim, noheaderread, noheadertrim, noinnertrim
Encoding: header line	ignoreencoding, interpretencoding
X-Envelope-to: Header Lines generation	nox_env_to, x_env_to
Return-path: header line generation	addreturnpath, noaddreturnpath
Envelope To: and From: Addresses in Received: Header Lines	noreceivedfor, noreceivedfrom, receivedfor, receivedfrom
Postmaster address	aliaspostmaster, noreturnaddress, noreturnpersonal, returnaddress, returnpersonal
Blank envelope return addresses	returnenvelope
Comments in address header lines	commentinc, commentmap, commentomit, commentstrip, commenttotal, sourcecommentinc, sourcecommentmap, sourcecommentomit, sourcecommentstrip, sourcecommenttotal
Personal names in address header lines	personalinc, personalmap, personalomit, personalstrip, sourcepersonalinc, sourcepersonalmap, sourcepersonalomit, sourcepersonalstrip
Alias file and alias database probes	aliaslocal
Subaddresses	subaddressexact, subaddressrelaxed, subaddresswild
Addresses produced by aliases	viaaliasoptional, viaaliasrequired
Two or four digit date conversion	datefour, datetwo
Day of week in date specifications	dayofweek, nodayofweek
Automatic splitting of long header lines	maxheaderaddrs, maxheaderchars
Header alignment and folding	headerlabelalign, headerlinelength
Automatic defragmentation of messages and partial messages	defragment, nodefragment
Automatic fragmentation of large messages	maxblocks, maxlines

Table 4-7 Channel Keywords Grouped by Functionality *(Continued)*

Functionality	Associated Keywords
Absolute message size limits	`blocklimit`, `linelimit`, `noblocklimit`, `nolinelimit`, `sourceblocklimit`
Maximum length header	maxprocchars
Mail delivery to over quota users	`holdexquota`, `noexquota`
Gateway daemons	daemon
Processing of account or message router mailbox	user
Message logging	`logging`, `nologging`
Debugging channel master and slave programs	`master_debug`, `nomaster_debug`, `noslave_debug`, `slave_debug`
Sensitivity checking	`sensitivitycompanyconfidential`, `sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate`
SASL configuration	`maysaslserver`, `mustsaslserver`, `nosasl`, `nosaslserver`, `nosasl`, `saslswitchchannel`
Verify the domain on mail From: is in the DNS	`mailfromdnsverify`, `nomailfromdnsverify`
Channel operation type	submit
Filter file location	`channelfilter`, `destinationfilter`, `fileinto`, `filter`, `nochannelfilter`, `nodestinationfilter`, `nofileinto`, `nofilter`, `nosourcefilter`, `sourcefilter`
Authenticated address from SMTP AUTH in header	authrewrite
Transport layer security	`maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, `tlsswitchchannel`
MS Exchange Gateway channels	`msexchange`, `nomsexchange`
Remove source routes	dequeue_removeroute
Default language	language
Loopcheck	`loopcheck`, `noloopcheck`
Service	`noservice`, `service`
Deferred delivery	`backoff`, `nonurgentbackoff`, `normalbackoff`, `urgentbackoff`
Lines over 1000 characters	`rejectsmtp`, `truncatesmtp`, `wrapsmtp`

Alias File

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

A physical line in this file is limited to 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>

user@domain: <address> <address> ...
```

The following is an example aliases file:

```
! A /var/mail user
mailsrv@siroe.com: mailsrv@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 alias 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 alias file; for example, it must be world readable.

The contents of the included file are inserted into the alias 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 alias file itself. Indeed, include files may themselves include other files. Up to three levels of include file nesting are allowed.

/var/mail Channel Option File

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 4-8 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.
`FORWARD_FORMAT` (string)	Specifies the location of the users' `.forward` files. In this string, %u is replaced by each user's id, and %h by each user's home directory. The default behavior, if this option is not explicitly specified, corresponds to: `FORWARD_FORMAT=%h/.forward`

Table 4-8 Local Channel Options *(Continued)*

Options	Descriptions
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 timeouts, the message is returned to the original sender with an error message resembling "Timeout waiting for *userb*'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

SMTP Channel Option Files

An option file may be used to control various characteristics of TCP / IP channels. Most of the options described actually relate to the SMTP protocol itself, rather than to the TCP / IP transport. As such, other MTA channels that use the SMTP protocol over other transports may have similar options.

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.

Note that while master channel programs (the outgoing / destination channel) read the global option file (*msg_svr_base*/config/option.dat) each time they run, the slave channel program reads the option file only when it is first started, and will not see changes until restarted.

For incoming messages, the TCP/IP channel options (in the SMTP channel options file, for example *msg_svr_base*/config/tcp_local_option) are options only for the incoming channel (slave channel program). These options are not to be used with other channels that might supposedly handle the incoming messages, like for example, channels with the *switchchannel keyword enabled.

Format of the File

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 floating point value, depending on the option's requirements. 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 *vb*.

Available SMTP Channel Options

The available options are listed in Table 4-9.

Table 4-9 SMTP Channel Options

Option	Description
522_PERMANENT_ERROR_STRING	Provides flexibility in handling 552 responses against broken SMTP servers. This option can be set to a list of 552 status strings that are to be treated as permanent errors. Multiple strings should be separated by vertical bars. The string needs to include the extended status code, assuming one is present, as well as the error text. The “552” should not be included.
ALLOW_ETRNS_PER_SESSION (integer)	Limits the number of ETRN commands accepted per session. The default is 1.
ALLOW_RECIPIENTS_PER_TRANSACTION (Integer)	Limits the number of recipients allowed per message. This option applies to the RCPT TO and SMTP VRFY commands. The default is 128.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
ALLOW_REJECTIONS_BEFORE_DEFERRAL (integer)	Set a limit on the number of bad RCPT TO: addresses that are allowed during a single session. That is, after the specified number of To: addresses have been rejected, all subsequent recipients, good or bad, are rejected with a 4xx error.
ALLOW_TRANSACTIONS_PER_SESSION (Integer)	Limits the number of messages allowed per connection. The default is no limit.
ATTEMPT_TRANSACTIONS_PER_SESSION (Integer)	Limits the number of messages the MTA attempts to transfer during any one connection session.
BANNER_ADDITION (U.S. ASCII String)	Adds the specified string to the SMTP banner line. The vertical bar character (|) is not permitted in the string.
BANNER_HOST (U.S. ASCII String)	Sets the host name that appears in the SMTP banners. The SMTP banners are the initial greetings given by the SMTP server and the HELO/EHLO commands issued by the SMTP client.
CHECK_SOURCE (0 or 1)	Controls whether or not the name found from a DNS lookup (or the IP domain literal, if DNS lookups have been disabled) is included in the constructed Received: header as a comment after the presented name when the determined name does not match the name presented by the remote SMTP client on the HELO or EHLO line. The SMTP server normally attempts to determine the name of the host from which a connection has been received, as specified by the `ident*` channel keywords. A value of 1 (default) enables the inclusion of the determined name when different from the presented name. A value of 0 disables the inclusion of any such comment thereby eliminating one of the more useful checks of message validity.
COMMAND_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive general SMTP commands (commands other than those with explicitly specified time-out values set using other specifically named options). The default value is 10.
COMMAND_TRANSMIT_TIME (Integer)	Specifies, in minutes, how long to spend transmitting general SMTP commands (commands other than those with explicitly specified time-out values set using other specifically named options). The default value is 10.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
CUSTOM_VERSION_STRING (U.S. ASCII String)	Overrides part of the default banner string that specifies product name and version number. This option is not recommended to be used.
DATA_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive data during an SMTP dialogue. The default is 5.
DATA_TRANSMIT_TIME (Integer)	Specifies, in minutes, how long to spend transmitting data during an SMTP dialogue. The default is 10.
DISABLE_ADDRESS (0 or 1)	The MTA SMTP server implements a private command XADR. This command returns information about how an address is routed internally by the MTA as well as general channel information. Releasing such information may constitute a breach of security for some sites. Setting the DISABLE_ADDRESS option to 1 disables the XADR command. The default is 0, which enables the XADR command.
DISABLE_CIRCUIT (0 or 1)	Enables or disables the private XCIR command implemented by the SMTP server. The XCIR command returns MTA circuit check information. Releasing such information may constitute a breach of security for some sites. Setting DISABLE_CIRCUIT to 1 disables the XCIR command. Setting DISABLE_CIRCUIT to 0 enables the XCIR command. If DISABLE_CIRCUIT is not explicitly set, then use of this XCIR command is controlled by the DISABLE_GENERAL option setting.
DISABLE_EXPAND (0 or 1)	The SMTP EXPN command is used to expand mailing lists. Exposing the contents of mailing lists to outside scrutiny may constitute a breach of security for some sites. The DISABLE_EXPAND option, when set to 1, disables the EXPN command completely. The default value is 0, which causes the EXPN command to work normally. Note that mailing list expansion can also be blocked on a list-by-list basis by setting the expandable attribute to False in the list's directory entry.
DISABLE_GENERAL (0 or 1)	Enables or disables the private XGEN command implemented by the SMTP server. The XGEN command returns status information about whether a compiled configuration and compiled character set are in use. Releasing such information may constitute a breach of security for some sites. Setting DISABLE_GENERAL to 1 disables the XGEN command. The default is 0, which enables the XGEN command.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
DISABLE_SEND (0 or 1)	Disable the SMTP SEND FROM:, SAML FROM:, and SOML FROM: commands. Setting this option to 1 disables the commands. The default is 1.
DISABLE_STATUS (0 or 1)	The MTA SMTP server implements a private command XSTA. This command returns status information about the number of messages processed and currently in the MTA channel queues. Releasing such information may consisted a breach of security for some sites. Setting the DISABLE_STATUS option to 1 disables the XSTA command. The default is 0, which enables the XSTA command.
DOT_TRANSMIT_TIME (Integer)	Specifies, in minutes, how long to spend transmitting the dot (.) terminating the data in an SMTP dialogue. The default is 10.
EHLO_ADDITION	Specifies an SMTP extension or extensions to advertise as part of the EHLO response. To specify multiple extensions, separate them with the vertical bar character (|).
HIDE_VERIFY (0 or 1)	The SMTP VRFY command can be used to establish the legality of an address before using it. This command has been abused by automated query engines in some cases. The HIDE_VERIFY option, when set to 1, tells the MTA not to return any useful information in the VRFY command result. The default value is 0, which causes VRFY to act normally. The vrfy* channel keywords may be used to control the MTA's behavior on a per-channel basis.
INITIAL_COMMAND	Specifies an initial SMTP command string for the SMTP client to send.
LOG_BANNER (0 or 1)	The LOG_BANNER option controls whether the remote SMTP server banner line is included in mail.log* file entries when the logging channel keyword is enabled for the channel. A value of 1 (the default) enables logging of the remote SMTP server banner line; a value of 0 disables it. LOG_BANNER also affects whether a remote SMTP banner line, if available, is included in bounce messages generated by the channel.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
LOG_CONNECTION (integer)	The LOG_CONNECTION option controls whether or not connection information, for example, the domain name of the SMTP client sending the message, is saved in mail.log file entries and the writing of connection records when the logging channel keyword is enabled for the channel. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given below: Bit-0 Value-1: When set, connection information is included in E and D log records. Bit-1 Value-2: When set, connection open, close, and fail records are logged by message enqueue and dequeue agents such as the SMTP clients and servers. Bit-2 Value-4: When set, I records are logged recording ETRN events. Where Bit 0 is the least significant bit. This channel option defaults to the setting of the global MTA option LOG_CONNECTION as set in the MTA option file. This channel option may be set explicitly to override on a per-channel basis the behavior requested by the global option.
LOG_TRANSPORTINFO (0 or 1)	The LOG_TRANSPORTINFO controls whether transport information, such as the sending and receiving side IP addresses and TCP ports, is included in mail.log file entries when the logging channel keyword is enabled for the channel. A value of 1 enables transport information logging. A value of 0 disables it. This channel option defaults to the setting of the global MTA option LOG_CONNECTION as set in the MTA option file.
MAIL_TRANSMIT_TIME (Integer)	Specifies, in minutes, the time to wait for the transmit to complete. The default is 10.
MAX_CLIENT_THREADS	An integer number indicating the maximum number of simultaneous outbound connections that the client channel program allows. Note that multiple processes may be used for outbound connections, depending on how you have channel-processing pools set up. This option controls the number of threads per process. The default if this option is not specified is 10.
MAX_A_RECORDS	Specifies the maximum number of A records that the MTA should try using when attempting to deliver a message. The default is no limit.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
MAX_J_ENTRIES	Specifies the maximum number of J mail.log* entries to write during a single SMTP connection session. The default is 10.
MAX_HELO_DOMAIN_LENGTH	Specifies the length limit of the argument accepted on the HELO, EHLO, and LHLO line. If a client sends a longer host name argument, that command is rejected. The default is no limit.
MAX_MX_RECORDS (Integer <=32)	Specifies the maximum number of MX records that the MTA should try using when attempting to deliver a message. The maximum value is 32, which is also the default.
PROXY_PASSWORD	Specifies the password to authenticate the SMTP proxy to the SMTP server to which the proxy intends to shuttle SMTP commands from a client. This value must match the MMP SmtpProxyPassword parameter.
RCPT_TRANSMIT_TIME (Integer)	Specifies, in minutes, the time to wait for the transmit to complete. The default is 10.
STATUS_DATA_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive the SMTP response to your sent data; that is, how long to wait to receive a 550 (or other) response to the dot-terminating-sent data. The default value is 10. See also the STATUS_DATA_RECV_PER_ADDR_TIME, STATUS_DATA_RECV_PER_BLOCK_TIME, and STATUS_DATA_RECV_PER_ADDR_PER_BLOCK_TIME options.
STATUS_DATA_RECV_PER_ADDR_TIME (Floating Point Value)	Specifies an adjustment factor for how long to wait to receive the SMTP response to your sent data based on the number of addresses in the MAIL TO command. This value is multiplied by the number of addresses and added to the base wait time (specified with the STATUS_DATA_RECV_TIME option). The default is 0.083333.
STATUS_DATA_RECV_PER_BLOCK_TIME (Floating Point Value)	Specifies an adjustment factor for how long to wait to receive the SMTP response to your sent data based on the number of blocks sent. This value is multiplied by the number of blocks and added to the base wait time (specified with the STATUS_DATA_RECEIVE_TIME option). The default is 0.001666.

Table 4-9 SMTP Channel Options *(Continued)*

Option	Description
STATUS_DATA_RECV_PER_ADDR_PER_BLOCK_TIME (Floating Point Value)	Specifies an adjustment factor for how long to wait to receive the SMTP response to your sent data based on the number of addresses (in the MAIL TO command) per number of blocks sent. This value is multiplied by the number of addresses per block and added to the base wait time (specified with the STATUS_DATA_RECEIVE_TIME option). The default is 0.003333.
STATUS_MAIL_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive the SMTP response to a sent MAIL FROM command. (Also corresponds to the time we wait for the initial banner line, and the time to wait to receive a response to a HELO, EHLO, or RSET command.) The default is 10.
STATUS_RCPT_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive the SMTP response to a sent RCPT TO command. The default value is 10.
STATUS_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive the SMTP response to general SMTP commands, (commands other than those with specified time out values set using other specifically named options). The default value is 10.
STATUS_TRANSMIT_TIME (Integer)	Specifies, in minutes, how long to spend transmitting the SMTP response to an SMTP command.
TRACE_LEVEL (0, 1, or 2)	This option controls whether TCP/IP level trace is included in debug log files. The default value is 0, meaning that no TCP/IP packet traces are included; a value of 1 tells the MTA to include TCP/IP packet traces in any debug log files; a value of 2 tells the MTA to include DNS lookup information as well as TCP/IP packet traces.
TRANSACTION_LIMIT_RCPT_TO	Affects the MTA's behavior once ALLOW_TRANSACTION_PER_SESSION has been exceeded. The default is 0, meaning that once ALLOW_TRANSACTION_PER_SESSION has been exceeded the MTA rejects subsequent transactions during hat same session at the MAIL FROM: command. If set to 1, the subsequent transactions are instead rejected at the RCPT TO: command.

Conversions

There are two broad categories of conversions in the MTA, controlled by two corresponding mapping tables and the MTA conversions file.

The first category is that of character set, formatting, and labelling conversions performed internally by the MTA. The application of such conversions is controlled by the CHARSET-CONVERSION mapping table.

The second category is that of conversions of message attachments using external, third-party programs and site-supplied procedures, such as document converters and virus scanners. The application of such conversions is controlled by the CONVERSIONS mapping table, and messages requiring such conversions are thereby routed through the MTA conversion channel, which in turn executes the site-specified external conversion procedure.

The MTA conversions file is used to specify the details of external CONVERSION table triggered conversions and to specify the details of some internal CHARSET-CONVERSION table triggered conversions.

Character Set Conversion and Message Reformatting Mapping

One very basic mapping table in the MTA 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 performed.

The MTA probes 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. The keywords provided are listed in Table 4-10.

Table 4-10 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
`Always`	Force conversion even when 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.
`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.

For more information on character set conversion and message reformatting mapping, see the *Sun Java System Messaging Server Administration Guide*.

Conversion File

Configuration of the conversion channel in the MTA configuration file (`imta.cnf`) is performed by default. With the rewrite rules from the default configuration, an address of the form `user@conversion.`*localhostname* or `user@conversion` is routed through the conversion channel, regardless of what the `CONVERSIONS` mapping states.

The actual conversions performed by the conversion channel are controlled by rules specified in the MTA conversion file. This is the file specified by the `IMTA_CONVERSION_FILE` option in the MTA tailor file. By default, this is the file *msg_svr_base*`/imta/conversions`.

The MTA conversion file is a text file containing entries in a format that is modeled after MIME Content-Type parameters. Each entry consists of one or more lines grouped together; each line contains one or more `name=`*value*`;` parameter clauses. Quoting rules conform to MIME conventions for Content-Type header line parameters. Every line except the last must end with a semicolon (;). A physical line in this file is limited to 1024 characters. You can split a logical line into multiple physical lines using the backslash (\) continuation character. Entries are terminated either by a line that does not end in a semicolon, one or more blank lines, or both.

The rule parameters currently provided are shown in Table 4-11. Parameters not listed in the table are ignored.

Table 4-11 Conversion Parameters

Parameter	Description
`COMMAND`	Command to execute to perform conversion. This parameter is required; if no command is specified, the entry is ignored.
`DELETE`	0 or 1. If this flag is set, the message part is deleted. (If this is the only part in a message, then a single empty text part is substituted.)
`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.

Table 4-11 Conversion Parameters *(Continued)*

Parameter	Description
DPARAMETER-SYMBOL-*n*	Content-disposition parameters to convert to environment variables if present; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to convert, as matched by an IN-DPARAMETER-NAME-*m* clause. Each DPARAMETER-SYMBOL-*n* is extracted from the Content-Disposition: parameter list and placed in an environment variable prior to executing the site-supplied program.
IN-A1-FORMAT	Input A1-format from enclosing message/rfc822 part.
IN-A1-TYPE	Input A1-type from enclosing message/rfc822 part.
IN-CHAN	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-CHANNEL	Synonym for IN-CHAN.
IN-DESCRIPTION	Input MIME Content-Description to match for conversion.
IN-DISPOSITION	Input MIME Content-Disposition to match for conversion.
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-DPARAMETER-NAME-*n*	Input MIME Content-Disposition parameter name whose value is to be checked; *n* = 0, 1, 2....
IN-DPARAMETER-VALUE-*n*	Input MIME Content-Disposition parameter value that must match corresponding IN-DPARAMETER-NAME (wildcards allowed). 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.
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-PARAMETER-NAME-*n*	Input MIME Content-Type parameter name whose value is to be checked; *n* = 0, 1, 2....
IN-PARAMETER-VALUE-*n*	Input MIME Content-Type parameter value that must match corresponding IN-PARAMETER-NAME (wildcards allowed). 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.
IN-SUBJECT	Input Subject from enclosing MESSAGE/RFC822 part.

Table 4-11 Conversion Parameters *(Continued)*

Parameter	Description
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-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.
MESSAGE-HEADER-FILE	Writes all, part, or none of the original headers of a message to the file specified by MESSAGE_HEADERS. If set to 1, the original headers of the immediately enclosing message part are written to the file specified by MESSAGE_HEADER. If set to 2, the original headers of the message as a whole (the outermost message headers) are written to the file.
ORIGINAL-HEADER-FILE	0 or 1. If set to 1, the original headers of the enclosing MESSAGE/RFC822 part are written to the file represented by the OUTPUT_HEADERS symbol.
OUT-CHAN	Output channel to match for conversion (wildcards allowed). The conversion specified by this entry is performed only if the message is destined for the specified channel.
OUT-CHANNEL	Synonym for OUT-CHAN.
OUT-DESCRIPTION	Output MIME Content-Description if it is different than the input MIME Content-Description.
OUT-DISPOSITION	Output MIME Content-Disposition if it is different than the input MIME Content-Disposition.
OUT-DPARAMETER-NAME-*n*	Output MIME Content-Disposition parameter name; *n*=0, 1, 2...
OUT-DPARAMETER-VALUE-*n*	Output MIME Content-Disposition parameter value corresponding to OUT-DPARAMETER-NAME-*n*.
OUT-MODE	Mode in which to read and store the converted file. This should be one of: BLOCK (binaries and executables) or TEXT.
OUT-ENCODING	Encoding to apply to the converted file when the message is reassembled.
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*.
OUT-SUBTYPE	Output MIME type if it is different than the input MIME type.
OUT-TYPE	Output MIME type if it is different than the input type.

Table 4-11 Conversion Parameters *(Continued)*

Parameter	Description
OVERRIDE-HEADER-FILE	0 or 1. If set, then MIME headers are read from the OUTPUT_HEADERS symbol, overriding the original headers in the enclosing MIME part.
OVERRIDE-OPTION-FILE	If set, the conversion channel reads options from the OUTPUT_OPTIONS environment variable.
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... Takes as argument the name of the MIME parameter to copy, as matched by an IN-PARAMETER-NAME-*n* clause.
PARAMETER-SYMBOL-*n*	Content-Type parameters to convert to environment variables if present; *n* = 0, 1, 2... Takes as argument the name of the MIME parameter to convert, as matched by an IN-PARAMETER-NAME-*n* clause. Each PARAMETER-SYMBOL-*n* is extracted from the Content-Type: parameter list and placed in an environment variable of the same name prior to executing the site-supplied program. Takes as the argument the variable name into which the MIME parameter to convert, as matched by an IN-PARAMETER-NAME-*n* clause.
PART-NUMBER	Dotted integers: *a. b. c*... The part number of the MIME body part.
RELABEL	0 or 1. This flag causes an entry to be ignored during conversion channel processing. However, if this flag is 1, then MIME header enabling is performed during character set conversions.
SERVICE-COMMAND	The command to execute to perform service conversion. The COMMAND parameter is required for conversion channel processing while SERVICE-COMMAND is an optional feature of charset conversion processing; if no command is specified, the entry is ignored. 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.
TAG	Input tag, as set by a mail list CONVERSION_TAG parameter.

Predefined Environment Variables

Table 4-12 shows the basic set of environment variables available for use by the conversion command.

Table 4-12 Environment Variables used by the Conversion Channel

Environment Variable	Description
ATTACHMENT_NUMBER	Attachment number for the current part.
CONVERSION_TAG	Current list of active conversion tags. This variable corresponds to the TAG conversion parameter.
INPUT_CHANNEL	Channel that enqueued the message to the conversion channel. This variable corresponds to the IN-CHANNEL conversion parameter.
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 headers for the enclosing 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 headers 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_CHANNEL	Channel to which the message is headed. This variable corresponds to the IN-CHANNEL conversion parameter.
OUTPUT_FILE	Name of the file where the site-supplied program should store its output. The site-supplied program should create and write this file.
OUTPUT_HEADERS	Name of the file where the site-supplied program should store MIME header lines for an enclosing part. The site-supplied program should create and write this file. Note that file should contain actual header lines (not option=value lines) followed by a blank line as its final line.
OUTPUT_OPTIONS	Name of the file from which the site-supplied program should read conversion channel options. Note that file should include header lines, followed by a blank line as its final line.
PART_NUMBER	Part number for the current part. Specified as dotted integers: *a. b. c...*
PART_SIZE	Size in bytes of the part being processed.

Additional environment variables containing Content-type: parameter information or Content-disposition: parameter information can be created as needed using the PARAMETER-SYMBOL-*n* or DPARAMETER-SYMBOL-*n* parameters respectively.

Table 4-13 displays additional override options available for use by the conversion channel. The converter procedure may use these to pass information back to the conversion channel. To set these options, set OVERRIDE-OPTION-FILE=1 in the desired conversion entry and then have the converter procedure set the desired options in the OUTPUT_OPTIONS file.

Table 4-13 Options for passing information back to the conversion channel

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 in the error text returned to the message 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/bin/msg/imtasdk/include/pmdf_err.h

Mapping File

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 just this sort of mapping table. The MTA database files, however, do not provide wildcard-lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

The mapping file provides the MTA with facilities for supporting multiple mapping tables. Full wildcard facilities are provided, and multistep 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 serve to eliminate the need for most of the entries in an equivalent database, and this may result in lower overhead overall.

For discussion on REVERSE and FORWARD address mapping, see the *Sun Java System Messaging Server Administration Guide*.

Locating and Loading the Mapping File

All mappings are kept in the MTA mapping file. (This is the file specified with the `IMTA_MAPPING_FILE` option in the MTA tailor file; by default, this is *msg_svr_base*`/config/mappings`.) The contents of the mapping file is incorporated into the compiled configuration.

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

File Format in the Mapping File

The mapping file consists of a series of separate tables. Each table begins with its name. Names always have an alphabetic character in the first column. The table name is followed by a required blank line, and then by the entries in the table. Entries consist of zero or more indented lines. Each entry must be preceded by at least one space. 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:

```
TABLE-1-NAME

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

TABLE-2-NAME

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

           .
           .
           .

TABLE-m-NAME

           .
           .
           .
```

An application using the mapping table `TABLE-2-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 may be continued by ending them with a backslash (\). The white space between the two columns and before the first column may not be omitted.

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

Including Other Files in the Mapping File

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

```
<file-spec
```

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

Mapping Operations

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

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

Mapping Entry Patterns

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

Table 4-14 Mapping Pattern Wildcards

Wildcard	Description
%	Match exactly one character.
*	Match zero or more characters, with maximal or “greedy” left-to-right matching

Table 4-14 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, that is, 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 4-14 Mapping Pattern Wildcards *(Continued)*

${IPv6}	Match an IPv6 address.

For more information about mapping pattern wildcards, see the section "Mapping File" in the chapter "About MTA Services and Configuration" in the *Sun Java System Messaging Server Administration Guide*.

Mapping Entry Templates

Table 4-15 lists the special substitution and standard processing metacharacters. Any other metacharacters are reserved for mapping-specific applications.

See the *Sun Java System Messaging Server Administration Guide* for more discussion on mapping entry templates.

Table 4-15 Mapping Template Substitutions and Metacharacters

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

Table 4-15 Mapping Template Substitutions and Metacharacters *(Continued)*

Substitution sequence	Substitutes
$\	Forces subsequent text to lowercase.
$^	Forces subsequent text to uppercase.
$_	Leaves subsequent text in its original case.
$=	Specifies that substituted characters undergo quoting appropriate for insertion into LDAP search filters.
$:*x*	Match only if the specified flag is set.
$;*x*	Match only if the specified flag is clear.

For more information on the substitution sequences and metacharacters, see the "About MTA Services and Configuration" chapter in the *Sun Java System Messaging Server Administration Guide*.

Option File

Global MTA options, as opposed to channel options, are specified in the MTA option file.

The MTA uses an option file to provide a means of overriding 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.

Locating and Loading the MTA Option File

The option file is the file specified with the `IMTA_OPTION_FILE` option in the IMTA tailor file (*msg_svr_base*`/config/imta_tailor`). By default, this is *msg_svr_base*`/config/option.dat`.

Option File Format and Available Options

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, an integer, or a floating point value depending on the option's requirements. 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*.

Long option values may be broken onto several lines. Each line that is to be continued should end with a back slash (\).

Comments are allowed. Any line that begins with an exclamation point (!), hash (#), or semicolon (;) is considered to be a comment and is ignored. You are allowed comments between continuation lines. Blank lines are also ignored in any option file.

The available options are listed in Table 4-16.

Table 4-16 Option File Options

Options	Description
ACCESS_ERRORS (Integer 0 or 1)	If ACCESS_ERRORS is set to 0 (the default), when an address causes an access failure the MTA reports it as an “illegal host or domain” error. This is the same error that would occur if the address were simply illegal. Although confusing, this usage provides an important element of security in circumstances where information about restricted channels should not be revealed. Setting ACCESS_ERRORS to 1 overrides this default and provide a more descriptive error.
ACCESS_ORCPT (0 or 1)	Setting ACCESS_ORCPT to 1 adds an additional vertical bar delimited field to the probe value passed to the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, and ORIG_MAIL_ACCESS mapping tables that contains the original recipient (ORCPT) address. If the message doesn't have an ORCPT address the original unmodified RCPT TO: address is used instead. The default is 0.

Table 4-16 Option File Options *(Continued)*

Options	Description
ALIAS_DOMAINS (Integer)	Controls the format of alias file and alias database lookups. This option takes a bit-encoded integer as its argument. The default value is 1, meaning that alias file and alias database lookups probe with only the local part (mailbox portion) of the address. Not that for addresses matching the local channel, such a probe is made even if bit 0 (value 1) is not set. Setting bit 1 (value 2) causes a probe to be made using the entire address (including the domain name). Setting bit 2 (value 4) causes a wildcard (*) probe to be made. If all bits are set, that is ALIAS_DOMAIN=7, then the order of the probes is to first probe with the entire address (the most specific check), next probe with a wildcard (*) local part plus the domain name, and finally probe with just the local part.
ALIAS_ENTRY_CACHE_NEGATIVE	Controls negative caching of alias entries. A nonzero value enables caching of alias match failures. A zero value disables it. Default: 0
ALIAS_ENTRY_CACHE_SIZE	Controls the size, in entries, of the alias cache. Default: 1000
ALIAS_ENTRY_CACHE_TIMEOUT	Controls the timeout, in seconds, of the alias cache. Default: 600
ALIAS_URL0 ALIAS_URL1 ALIAS_URL2 ALIAS_URL3 (URL)	Specifies a URL to query for alias lookups. The URL must be specified using standard LDAP URL syntax, except the LDAP server and port must be omitted. The LDAP server and port are specified via the LDAP_HOST and LDAP_PORT options. See “MTA Direct LDAP Operation” in the *Sun Java System Messaging Server Administration Guide* for certain substitution sequences.
ALIAS_HASH_SIZE (Integer <= 32,767)	Sets the size of the alias hash table. This is an upper limit on the number of aliases that can be defined in the alias file. The default is 256; the maximum value is 32,767.
ALIAS_MAGIC	Determines he exact alias sources that are checked and the order in which they are checked. When set to 8764, 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 specified by the ALIAS_URL2 MTA option, and finally, the alias file. The alias database is not checked when this setting is active.
ALIAS_MEMBER_SIZE (Integer <= 20,000)	Controls the size of the index table that contains the list of alias translation value pointers. The total number of addresses on the right sides of all of the alias definitions in the alias file cannot exceed this value. The default is 320; the maximum value is 20,000.
ALLOW_UNQUOTED_ADDRS_VIOLATE_RFC2798	If set to 1, it will add additional filter terms to search on the syntactically invalid dequoted form of quoted addresses. Default: 0

Table 4-16 Option File Options *(Continued)*

Options	Description
BLOCK_LIMIT (Integer > 0)	Places an absolute limit on the size, in blocks, of any message that may be sent or received with the MTA. Any message exceeding this size is rejected. By default, the MTA imposes no size limits. Note that the `blocklimit` channel keyword can be used to impose limits on a per-channel basis. The size in bytes of a block is specified with the BLOCK_SIZE option.
BLOCK_SIZE (Integer > 0)	The MTA uses the concept of a "block" in several ways. For example, the MTA log files (resulting from placing the `logging` keyword on channels) record message sizes in terms of blocks. Message size limits specified using the `maxblocks` keyword are also in terms of blocks. Normally, an MTA block is equivalent to 1024 characters. This option can be used to modify this sense of what a block is. Caution: Reducing BLOCK_SIZE too much (to a value of 1) may have negative impact on the MTA.
BOUNCE_BLOCK_LIMIT (Integer)	Used to force bounces of messages over the specified size to return only the message headers, rather than the full message content.

BRIGHTMAIL_ACTION_*n*	As a pair with the matching `Brightmail_verdict_`*n* option, this can specify a Sieve command with optional if-then-else statement* to execute. For example, if you want to reject spam, then you may have the pair:
	
	`Brightmail_verdict_0=spamfolder`
	`Brightmail_action_0=data:,require "reject"; reject "Rejected by Brightmail";`
	
	The template for the Sieve command is:
	`data:,[require "`*command*`";] `*command*`;`
	Where the `require` statement is needed for `reject` and `fileinto`. Another example:
	
	`Brightmail_verdict_1=spam-folder`
	`Brightmail_action_1_=data;,require "fileinto";fileinto "Junk";`
	
	This files the spam (assuming `spam-folder` is the verdict returned by Brightmail for spam) into a folder called `Junk`. Without `Junk`, spam will be filed into the folder called `spam-folder`.
	
	Default: none

BRIGHTMAIL_CONFIG_FILE (path)	Required to activate Brightmail. Full file path and name of the Brightmail configuration file. When specified along with Brightmail_library, the MTA is enabled for Brightmail integration. Can also be used with SpamAssassin. Example: `/opt/mailwall/config` Default: None

Table 4-16 Option File Options *(Continued)*

Options	Description
BRIGHTMAIL_LIBRARY (path)	Required to activate Brightmail. Full file path and name of the Brightmail SDK shared library. When specified along with Brightmail_config_file, this library is loaded by the MTA at run time. Can also be used with SpamAssassin. Example: /opt/mailwall/lib/libbmiclient.so Default: None
BRIGHTMAIL_NULL_ACTION	Specifies a Sieve command with optional if-then-else statement* to execute when the verdict from Brightmail matches the Null action in the Brightmail configuration file. For example, if the Brightmail configuration file has: blSWClientDestinationLocal: spam| where the null or nothing after the | means the null action. If the verdict for the message is spam, matching the word spam before the |, then the null action will be taken by the MTA. There is usually no reason to specify this option, since the default action is discard, matching what Brightmail means by the null action. Can also be used with SpamAssassin. The template for the Sieve command is: data:,[require "*command*";] *command;* Where the require statement is needed for reject and fileinto. Default: data:,discard;
BRIGHTMAIL_OPTIONAL	If set to 1, when the MTA calls an initialization routine to load the Brightmail SDK, but fails, the MTA continues as if Brightmail was not enabled. This setting has no effect if the MTA is already in a dialogue with Brightmail and Brightmail dies. In this case the MTA returns a temporary error to the SMTP client. Default: 0
BRIGHTMAIL_STRING_ACTION	Specifies a Sieve command with optional if-then-else statement* to execute when the Brightmail verdict matches an action which is a string in the Brightmail configuration file. Can also be used with SpamAssassin. For example, if in the Brightmail configuration file you have blSWClientDestinationLocal: spam|spam-folder then spam-folder is a string. If the verdict is spam, then we have a string which matches the verdict. This option is rarely used, since the default action when a string is specified is to file the message into that folder. The template for the Sieve command is: data:,[require "*command*";] *command;* Where the require statement is needed for reject and fileinto. Default: data:,require "fileinto"; fileinto "$U"; $U is the string to the right of | in the blSWClientDestinationLocal value (in the example above it would be spam-folder)

Table 4-16 Option File Options *(Continued)*

Options	Description
BRIGHTMAIL_VERDICT_*n*	Brightmail_verdict_*n* and Brightmail_action_*n* are matched pairs, where *n* is a number from 0 to 9. These options are not normally specified if you take the default interpretation of Brightmail verdicts. The possible values for this option are specified by the values on the right of the | in the Brightmail configuration file options blSWClientDestinationLocal (for local domains) or blSWClientDesintationForeign (for domains that are not local). Using the example: blSWClientDestinationLocal=spam|spamfolder you would want to have Brightmail_verdict_0=spamfolder (not spam, which is to the left of |. This may seem non-intuitive, but that is indeed how it works. Default: none
CACHE_DEBUG (0 or 1)	When set to 1, this option indicates to various MTA components to write information about the domain, alias, and reverse caches to its debug log file just prior to exiting.
CHANNEL_TABLE_SIZE (Integer <= 32,767)	Controls the size of the channel table. The total number of channels in the configuration file cannot exceed this value. The default is 256; the maximum is 32,767.
CIRCUITCHECK_COMPLETED_BINS (comma separated list of up to eight integers)	Specifies the bin divisions, in seconds, for the MTA circuit check counters. The default values are 120, 300, 900, 1800, 3600, 7200, 14400, and 28800 (2 minutes, 5 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, and 8 hours respectively).
CIRCUITCHECK_PATCHS_SIZE (Integer <=256)	Controls the size of the circuit check paths table, and thus the total number of circuit check configuration file entries. The default is 10.
COMMENT_CHARS (Integer list)	Sets the comment characters in the MTA configuration files. The value of this option takes the form of a list of ASCII character values in decimal. The default is the list {33, 59}, which specifies exclamation points and semicolons as comment introduction characters.
CONTENT_RETURN_BLOCK_LIMIT (Integer)	Specifies the maximum size of an originating message that will be returned in a notification message. If the original message content is larger than this size, then the message will not be returned in a notification message. The units are in blocks (see BLOCK_SIZE).
CONVERSION_SIZE (Integer <= 2000)	Controls the size of the conversion entry table, and thus the total number of conversion file entries cannot exceed this number. The default is 32.

Table 4-16 Option File Options *(Continued)*

Options	Description
DEFER_GROUP_PROCESSING	Whether mail groups are expanded online (by the enqueuing tcp_smtp_server, for instance, or offline by enqueuing the group address unchanged to the reprocess channel, is controlled by the value of the mailDeferprocessing attribute on the LDAP entry for the group. If that attribute is absent, then the behavior of the system is controlled by the DEFER_GROUP_PROCESSING option. If this value is set, mail groups with no mailDeferProcessing attribute are expanded offline. Otherwise they are expanded online. The default is 1 (yes).
DELIVERY_OPTIONS	Controls the conversion of the mailDeliveryOption attribute into appropriate addresses. 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. Default: 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+$D@bitbucket
DEQUEUE_DEBUG (0 or 1)	Specifies whether debugging output from the MTA's dequeue facility (QU) is produced. If enabled with a value of 1, this output is produced on all channels that use the QU routines. The default of 0 disables this output.
DEQUEUE_MAP (0 or 1)	Determines whether or not a message is mapped into memory when dequeuing. The default is 1.
DOMAIN_FAILURE	Specifies a template to use in the event of a domain lookup failure. Default: reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temporary lookup failure
DOMAIN_HASH_SIZE (Integer <= 32,767)	Controls the size of the domain rewrite rules hash table. Each rewrite rule in the configuration file consumes one slot in this hash table; thus the number of rewrite rules cannot exceed this option's value. The default is 512; the maximum number of rewrite rules is 32,767.
DOMAIN_MATCH_CACHE_SIZE	Sets the maximum size of the MTA's private domain match cache. Default: 10000

Table 4-16 Option File Options *(Continued)*

Options	Description
DOMAIN_MATCH_CACHE_TIMEOUT	Sets the timeout for entries in the MTA's private domain match cache. Default: 600 (seconds)
DOMAIN_MATCH_URL	Sets the URL for vanity domain checking. The value of this option should be set to: ldap:///$B?msgVanityDomain?sub?(msgVanityDomain=$D)
DOMAIN_UPLEVEL (Integer, 0-3)	Controls the MTA domain and email address lookup. It accepts a two-bit binary value (0-3). Bit 0 (least significant bit) controls domain lookup. Bit 1 (most significant bit) controls address lookup. Table 4-17 describes this control in detail. For example, if the MTA is performing a domain lookup for desert.island.siroe.com, a DOMAIN_UPLEVEL value of 1 or 3 specifies a lookup for desert.island.siroe.com, island.siroe.com, siroe.com, and com. A value of 0 or 2 specifies a lookup for only desert.island.siroe.com. Similarly, if the MTA is performing an address lookup for rcrusoe@desert.island.siroe.com, then a DOMAIN_UPLEVEL value of 2 or 3 specifies that the MTA search for rcrusoe@desert.island.siroe.com, rcrusoe@island.siroe.com, rcrusoe@siroe.com, or a domain alias for rcrusoe@desert.island.siroe.com. A DOMAIN_UPLEVEL value of 0 or 1 specifies that the MTA search only for rcrusoe@desert.island.siroe.com. Default is 0.
EXPANDABLE_DEFAULT (Integer 0 or 1)	Specifies whether or not lists are expandable by default. The option, if set to 1 enables the SMTP EXPN command. 1 is the default and allows for mail list expansion.
EXPROUTE_FORWARD (Integer 0 or 1)	Controls the application of the exproute channel keyword to forward-pointing (To, Cc, and Bcc lines) addresses in the message header. A value of 1 is the default and specifies that exproute should affect forward pointing header addresses. A value of 0 disables the action of the exproute keyword on forward pointing addresses.
FILE_MEMBER_SIZE	Specifies the maximum size of the table that tracks the list of files contributed to the configuration.
FILTER_DISCARD (1 or 2)	Controls whether mailbox filter discard actions cause such discarded messages to be immediately discarded, or cause such messages to go to the FILTER_DISCARD channel. The FILTER_DISCARD channel keeps messages for a short period before discarding them. The default is FILTER_DISCARD=1, which means that messages discarded by a mailbox filter are immediately discarded. Setting FILTER_DISCARD=2 causes discarded messages to instead be routed to the filter_discard channel.

Table 4-16 Option File Options *(Continued)*

Options	Description
HEADER_LIMIT (Integer)	Specifies a maximum header size. If the message's header exceeds this limit, the message is rejected. The default is 2000.
HISTORY_TO_RETURN (1-200)	Controls how many delivery attempt history records are included in returned messages. The delivery history provides an indication of how many delivery attempts were made and might indicate the reason the delivery attempts failed. The default value for this option is 20.
HELD_SNDOPR (Integer 0 or 1)	Controls the production of operator messages when a message is forced into a held state because it has too many Received: header lines. The default is 0 and specifies that the syslog messages are not generated when messages or forced to .HELD status due to too many Received: header lines. The value of 1 specifies that syslog messages are generated.
HOST_HASH_SIZE (Integer <= 32,767)	Controls the size of the channel hosts hash table. Each channel host specified on a channel definition in the MTA configuration file (both official hosts and aliases) consumes one slot in this hash table, so the total number of channel hosts cannot exceed the value specified. The default is 512; the maximum value allowed is 32,767.
ID_DOMAIN (U.S. ASCII String)	Specifies the domain name to use when constructing message IDs. By default, the official host name of the local channel is used.
IMPROUTE_FORWARD (Integer 0 or 1)	Controls the application of the improute channel keyword to forward-pointing (To, Cc, and Bcc lines) addresses in the message header. A value of 1 is the default and specifies that improute should affect forward-pointing header addresses. A value of 0 disables the action of the improute keyword on forward-pointing addresses.
LDAP_ADD_HEADER	Specifies the LDAP attribute to use to specify header field values that are to be added to the message header if it is present. Typically, this option is not set because the default value mgrpAddHeader corresponds to the standard schema.
LDAP_ALIAS_ADDRESSES	Can be used to override the use of the mailAlternateAddress attribute.
LDAP_ATTR_DOMAIN_SEARCH_FILTER	Specifies the LDAP attribute name in the global Sun ONE LDAP Schema, v2 domain search template which contains the domain search pattern. This option is ignored if the LDAP_GLOBAL_CONFIG_TEMPLATE is not set. The default value of this option is inetDomainSearchFilter.
LDAP_ATTR_DOMAIN1_SCHEMA2	Specifies the LDAP attribute name for the primary domain attribute used by Sun ONE LDAP Schema, v2. The default value is sunPreferredDomain.
LDAP_ATTR_DOMAIN2_SCHEMA2	This is the LDAP attribute name for the secondary domain attribute used by Sun ONE LDAP Schema, v2. The default value is associatedDomain.
LDAP_ATTR_MAXIMUM_MESSAGE_SIZE	Specifies the LDAP attribute to use to specify the maximum message size in bytes that can be sent to the group. Typically, this option is not set because the default value mgrpMsgMaxSize corresponds to the standard schema.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_AUTH_DOMAIN	Specifies the LDAP attribute to use to identify domains (including subdomains) from which users are allowed to send messages to the mail group. Typically, this option is not set because the default value mgrpAllowedDomain corresponds to the standard schema.
LDAP_AUTH_PASSWORD	Specifies the LDAP attribute to use to specify a password needed to post to the list. Typically, this option is not set because the default value mgrpAuthPassword corresponds to the standard schema.
LDAP_AUTH_POLICY	Specifies the LDAP attribute to use to specify the level of authentication required to access the list of broadcaster addresses. Typically, this option is not set because the default value mgrpBroadcasterPolicy corresponds to the standard schema.
LDAP_AUTH_URL	Specifies the LDAP attribute to use to identify mail users allowed to send messages to the mail group. Typically, this option is not set because the default value mgrpAllowedBroadcaster corresponds to the standard schema.
LDAP_BLOCKLIMIT	Specifies the LDAP attribute to use to impose a size limit in units of MTA blocks that can be sent to this user or group. Typically, this option is not set because the default value mailMsgMaxBlocks corresponds to the standard schema.
LDAP_CANT_DOMAIN	Specifies the LDAP attribute to use to identify domains from which users are not allowed to send messages to the mail group. Typically, this option is not set because the default value mgrpDisallowedDomain corresponds to the standard schema.
LDAP_CANT_URL	Specifies the LDAP attribute to use to identify mail users not allowed to send messages to the mail group. Typically, this option is not set because the default value mgrpDisallowedBroadcaster corresponds to the standard schema.
LDAP_CAPTURE	Specifies the attribute used to specify one or more message capture addresses. No default.
LDAP_CONVERSION_TAG	Specifies the LDAP attribute to use for conversion tags attached to a message to this user or group. Tag-specific conversion actions are specified in the MTA configuration. Typically, this option is not set because the default value mailConversionTag corresponds to the standard schema.
LDAP_DEFAULT_ATTR	Specifies the default attribute if no attribute is specified in the LDAP query for URLs that are supposed to return a single result.
LDAP_DEFAULT_DOMAIN	Overrides the service.defaultdomain configutil parameter.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_DELIVERY_FILE	Specifies the LDAP attribute to use to specify the fully-qualified local path of the file to which all messages sent to the mailing list are appended. Typically, this option is not set because the default value mailDeliveryFileURL corresponds to the standard schema
LDAP_DELIVERY_OPTION	Specifies the LDAP attribute to use to specify delivery options for the mail recipient. Typically, this option is not set because the default value mailDeliveryOption corresponds to the standard schema.
LDAP_DISK_QUOTA	Specifies the LDAP attribute to use to specify disk space allowed for the user's mailbox in bytes. Typically, this option is not set because the default value mailQuota corresponds to the standard schema.
LDAP_DOMAIN_ATTR_ALIAS	Specifies the LDAP attribute name which contains a pointer to another domain node in Sun ONE LDAP Schema, v1. The default value is aliasedObjectName.
LDAP_DOMAIN_ATTR_AUTOREPLY_TIMEOUT	No default.
LDAP_DOMAIN_ATTR_BASEDN	Specifies the LDAP attribute name which contains the baseDN of the user subtree associated with a given domain in Sun ONE LDAP Schema, v1. In Sun ONE LDAP Schema, v2 mode, this attribute specifies the canonical organization node (under which the users are located) pointed to by a domain index node. The default value is inetDomainBaseDN.
LDAP_DOMAIN_ATTR_BLOCKLIMIT	Specifies the LDAP attribute to use to impose a size limit in units of MTA blocks on all messages sent to addresses in this domain. Typically, this option is not set because the default value mailDomainMsgMaxBlocks corresponds to the standard schema.
LDAP_DOMAIN_ATTR_CANONICAL	Specifies the LDAP attribute name which contains the canonical domain name associated with a Sun ONE LDAP Schema, v1 domain entry. The default value is inetCanonicalDomainName.
LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS	Specifies the LDAP attribute to use to specify an address to be substituted for any address in the domain that does not match any user or group in the domain. Typically, this option is not set because the default value mailDomainCatchallAddress corresponds to the standard schema.
LDAP_DOMAIN_ATTR_CONVERSION_TAG	Specifies the LDAP attribute to use for one or more conversion tags attached to messages to any user in the domain. Tag-specific conversion actions are specified in the MTA configuration. Typically, this option is not set because the default value mailDomainConversionTag corresponds to the standard schema.
LDAP_DOMAIN_ATTR_DISK_QUOTA	No default.
LDAP_DOMAIN_ATTR_FILTER	Specifies the LDAP attribute to use to specify the sieve filter for all users in the domain. Typically, this option is not set because the default value mailDomainSieveRuleSource corresponds to the standard schema.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_DOMAIN_ATTR_MAIL_STATUS	Specifies the LDAP attribute to use to specify the mail status. Typically, this option is not set because the default value `mailDomainStatus` corresponds to the standard schema.
`LDAP_DOMAIN_ATTR_MESSAGE_QUOTA`	No default.
`LDAP_DOMAIN_ATTR_OPTIN` (ASCII)	The name of the LDAP attribute used to activate Brightmail on a per-domain basis. It applies to the destination domain. It is just like LDAP_optin above except it should be in the objectclass `mailDomain`. Default: none
`LDAP_DOMAIN_ATTR_PRESENCE`	No default.
`LDAP_DOMAIN_ATTR_RECIPIENTCUTOFF`	No default.
`LDAP_DOMAIN_ATTR_RECIPIENTLIMIT`	No default.
`LDAP_DOMAIN_ATTR_REPORT_ADDRESS`	Specifies the LDAP attribute to use to specify the header From: address in DSNs reporting problems associated with recipient addresses in the domain. It is also used when reporting problems to users within the domain regarding errors associated with non-local addresses. If this attribute is not set, the reporting address defaults to “postmaster@domain.” Typically, this option is not set because the default value `mailDomainReportAddress` corresponds to the standard schema.
`LDAP_DOMAIN_ATTR_ROUTING_HOSTS`	Specifies the LDAP attribute to use to specify the fully-qualified host name of the MTA responsible for making routing decisions for users in this (and all contained) domain(s). Typically, this option is not set because the default value `mailRoutingHosts` corresponds to the standard schema.
`LDAP_DOMAIN_ATTR_SOURCEBLOCKLIMIT`	No default.
`LDAP_DOMAIN_ATTR_SMARTHOST`	Specifies the LDAP attribute to use to specify the fully-qualified host name of a mail server responsible for handling mail for users not found in the local directory. Typically, this option is not set because the default value `mailRoutingSmarthost` corresponds to the standard schema.
`LDAP_DOMAIN_ATTR_STATUS`	Specifies the LDAP attribute to use to specify the current status of the mail domain (`active`, `inactive`, `deleted`, or `hold`). Typically, this option is not set because the default value `mailDomainStatus` corresponds to the standard schema.
`LDAP_DOMAIN_ATTR_UID_SEPARATOR`	Specifies the LDAP attribute to use to override the default mailbox (MB) home. Typically, this option is not set because the default value `domainUidSeparator` corresponds to the standard schema.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_DOMAIN_FILTER_SCHEMA1	This is the LDAP search filter used for Sun ONE LDAP Schema, v1 domain lookups. Default: (|(objectclass=inetDomain)(objectclass=inetdomainalias)).
LDAP_DOMAIN_FILTER_SCHEMA2	This is the LDAP search filter used for Sun ONE LDAP Schema, v2 domain lookups. Default: (objectclass=sunManagedOrganization)
LDAP_DOMAIN_ROOT	If set, overrides the service.dcroot configutil parameter.
LDAP_DOMAIN_TIMEOUT	Controls the retention time for entries in the domain map cache. This value is expressed in seconds. Default: 60 * 15 (or 15 minutes)
LDAP_END_DATE	Specifies the vacation end date attribute. Default: vacationEndDate
LDAP_EQUIVALENCE_ADDRESSES	Can be used to override the mailEquivalentAddress attribute.
LDAP_ERRORS_TO	Specifies the LDAP attribute to use to specify the recipient of error messages generated when messages are submitted to this list. Typically, this option is not set because the default value mgrpErrorsTo corresponds to the standard schema.
LDAP_EXPANDABLE	Specifies the attribute to check for group expansion as part of an SMTP EXPN command. Default: mgmanMemberVisibility
LDAP_FILTER	Specifies the LDAP attribute to use to specify SIEVE rules for filtering mail. Typically, this option is not set because the default value mailSieveRuleSource corresponds to the standard schema.
LDAP_FORWARDING_ADDRESS	Default: mailForwardingAddress
LDAP_GLOBAL_CONFIG_TEMPLATES	Specifies the LDAP baseDN which contains the global Sun ONE LDAP Schema, v2 domain template for domain searches. Use of this option is not recommended. There is no default value.
LDAP_GROUP_DN	Specifies the LDAP attribute to use to identify a member of a group of names where each name was given a uniqueIdentifier to ensure its uniqueness. Typically, this option is not set because the default value uniqueMember corresponds to the standard schema.
LDAP_GROUP_MAIL_STATUS	Controls mail-specific group status attributes.
LDAP_GROUP_OBJECT_CLASSES	Specifies different sets of object classes for groups.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_GROUP_RFC822	Specifies the LDAP attribute to use to identify recipients of mail sent to mail group. Typically, this option is not set because the default value mgrpRFC822MailMember (or rfc822MailMember, used for backward compatibility) corresponds to the standard schema.
LDAP_GROUP_STATUS	Used to select alternate general status attributes for groups.
LDAP_GROUP_URL1	Specifies the LDAP attribute to use as an alternative method of specifying mail group membership. Typically, this option is not set because the default value mgrpDeliverTo corresponds to the standard schema.
LDAP_GROUP_URL2	Specifies the LDAP attribute to use to specify a list of URLs, which, when expanded, provides a list of mailing list member addresses. Typically, this option is not set because the default value memberURL corresponds to the standard schema.
LDAP_HASH_SIZE	Specifies the size of the internal table of LDAP attribute names.
LDAP_HOST (Host name)	If set, overrides the MTA's use of the local.ugldaphost configutil parameter in accessing the LDAP directory server.
LDAP_HOST_ALIAS_LIST	If set, overrides the MTA's use of the local.imta.hostnamealiases configutil parameter in accessing the LDAP directory server.
LDAP_LOCAL_HOST	If set, overrides the MTA's use of the local.hostname configutil parameter in accessing the LDAP directory server.
LDAP_MAIL_REVERSES	Specifies the list of attributes to search containing addresses that are candidates for address reversal. 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.
LDAP_MAILHOST	Specifies the LDAP attribute to use to specify the fully-qualified host name of the MTA that is the final destination of messages sent to this recipient. Typically, this option is not set because the default value mailhost corresponds to the standard schema.
LDAP_MESSAGE_QUOTA	Specifies the LDAP attribute to use to specify the maximum number of messages permitted for a user. Typically, this option is not set because the default value mailMsgQuota corresponds to the standard schema.
LDAP_MODERATOR_URL	Specifies the LDAP attribute to use to specify the LDAP URI or mailto URL identifying the moderators allowed to submit messages to this list. Typically, this option is not set because the default value mgrpModerator corresponds to the standard schema.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_OPTIN (ASCII)	The name of the LDAP attribute used to activate Brightmail on a per-user basis. This should be an attribute in the inetMailUser objectclass. If you do not have another predefined attribute, use mailAntiUBEService. The attribute itself (example: mailAntiUBEService) is multi-valued, case-sensitive. Its value could be either spam or virus in lowercase. If the user is opting for both, then he would have two such attributes, one containing spam, one containing virus. Default: none
LDAP_PASSWORD	If set, overrides the MTA's use of the local.ugldapbindcred configutil parameter in accessing the LDAP directory server.
LDAP_PERSONAL_NAME	No default.
LDAP_PORT (Integer)	Specifies the port to which to connect when performing LDAP queries. The default value is 389, the standard LDAP port number.
LDAP_PREFIX_TEXT	Specifies the LDAP attribute to use to specify the text to be added to the beginning of the message text. Typically, this option is not set because the default value mgrpMsgPrefixText corresponds to the standard schema.
LDAP_PRESENCE	Specifies 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.
LDAP_PRIMARY_ADDRESS	Overrides the LDAP attribute to use to specify the primary address typically stored in the mail attribute.
LDAP_PROGRAM_INFO	Specifies the LDAP attribute to use to specify one or more programs used for program delivery. Typically, this option is not set because the default value mailProgramDeliveryInfo corresponds to the standard schema.
LDAP_RECIPIENTLIMIT	No default.
LDAP_RECIPIENTCUTOFF	No default.
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. Default: mgrpMsgRejectAction
LDAP_REJECT_TEXT	Specifies the LDAP attribute to use to specify text to return if any of the authentication attributes cause the message to be rejected. Typically, this option is not set because the default value mailRejectText corresponds to the standard schema.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_REMOVE_HEADER	Specifies the LDAP attribute to use to specify a header field that is to be removed from the message header if it is present. Typically, this option is not set because the default value mgrpRemoveHeader corresponds to the standard schema.
LDAP_REPROCESS	Specifies the attribute used for deferred mail processing. Default: mailDeferProcessing
LDAP_ROUTING_ADDRESS	Specifies the LDAP attribute to use to determine whether or not the address should be acted on at this time or forwarded to another system. Typically, this option is not set because the default value mailRoutingAddress corresponds to the standard schema.
LDAP_SCHEMALEVEL	If set to value 2, this enables support for Sun ONE LDAP Schema, v2.
LDAP_SCHEMATAG	Can be used to override the setting of the local.imta.schematag configutil parameter specifically for the MTA.
LDAP_SOURCEBLOCKLIMIT	No default.
LDAP_SPARE_1	Spare slot for additional attributes to be used to build customized address expansion facilities. No default.
LDAP_SPARE_2	Spare slot for additional attributes to be used to build customized address expansion facilities. No default.
LDAP_SPARE_3	Spare slot for additional attributes to be used to build customized address expansion facilities. No default.
LDAP_SPARE_4	Spare slot for additional attributes to be used to build customized address expansion facilities. No default.
LDAP_SPARE_5	Spare slot for additional attributes to be used to build customized address expansion facilities. No default.
LDAP_START_DATE	Specifies the vacation start date attribute. Default: vacationStartDate
LDAP_SUFFIX_TEXT	Specifies the LDAP attribute to use to specify the text to append to the text message. Typically, this option is not set because the default value mgrpMsgSuffixText corresponds to the standard schema.
LDAP_TIMEOUT (Integer)	Controls the length of time to wait (in hundredths of seconds) before timing out on an LDAP query. The default value is 180000.

Table 4-16 Option File Options *(Continued)*

Options	Description
LDAP_UG_FILTER	Object class settings are 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.
LDAP_UID	Specifies the LDAP attribute to use to identify the entry's userid. Typically, this option is not set because the default value uid corresponds to the standard schema.Identifies the entry's userid.
LDAP_USE_ASYNC	Controls the use of asynchronous LDAP lookups. 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. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in Table 4-20. Default: 0 (asynchronous LDAP looks are disabled)
LDAP_USERNAME	If set, overrides the MTA's use of the local.ugldapbinddn configutil parameter in accessing the LDAP directory server.
LDAP_USER_MAIL_STATUS	Controls mail-specific user status attributes.
LDAP_USER_OBJECT_CLASSES	Specifies different sets of object classes for users.
LDAP_USER_ROOT	If set, overrides the MTA's use of the local.ugldapbasedn configutil parameter.
LDAP_USER_STATUS	Used to select alternate general status attributes for users.
LINE_LIMIT (Integer)	Places an absolute limit on the overall number of lines in any message that may be sent or received with the MTA. Any message exceeding this limit is rejected. By default, the MTA imposes no line-count limits.The linelimit channel keyword can be used to impose limits on a per channel basis.
LINES_TO_RETURN (Integer)	Controls how many lines of message content the MTA includes when generating a notification message for which it is appropriate to return on a sample of the contents. Setting the LINES_TO_RETURN option to 0 disables partial content return. Only the headers of the message part are returned. The default is 20.

Table 4-16 Option File Options *(Continued)*

Options	Description
LOG_CONNECTION (Integer)	The LOG_CONNECTION option controls whether or not connection information, for example, the domain name of the SMTP client sending the message, is saved in mail.log file entries and the writing of connection records when the logging channel keyword is enabled for the channel. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given below: Bit-0 Value-1: When set, connection information is included in E and D log records. Bit-1 Value-2: When set, connection open, close, and fail records are logged by message enqueue and dequeue agents such as the SMTP clients and servers. Bit-2 Value-4: When set, I records are logged recording ETRN events. Where Bit 0 is the least significant bit. This channel option defaults to the setting of the global MTA option LOG_CONNECTION as set in the MTA option file. This channel option may be set explicitly to override on a per-channel basis the behavior requested by the global option.
LOG_CONNECTIONS_SYSLOG (0 or 1)	Sends MTA connection log file entries to syslog (UNIX) or event log (Windows NT). 0 is the default and indicates that syslog (event log) logging is not performed. A value of 1 indicates that syslog logging is performed.
LOG_SENSITIVITY (0 or 1)	Controls whether message Sensitivity: header values are included in log entries. A value of 1 enables such logging. The default value of 0 disables such logging. If logging is enabled, the sensitivity value is logged in an integer representation after the connection information and before the transport information.
LOG_DELAY_BINS	Specifies the bins for delivery delay range counters. The parameters for this options should be a comma-separated list of up to five integers. The default values are 60, 600, 6000, 60000, 600000.
LOG_ENVELOPE_ID (0 or 1)	Controls whether or not envelope IDs are logged. If envelope IDs are logged, they appear just before the message IDs in the log. The value of this option defaults to 0, which prevents envelope IDs from being logged.
LOG_FILENAME (0 or 1)	Controls whether the names of the files in which messages are stored are saved in the mail.log file. A value of 1 enables file name logging. A value of 0 (the default) disables it.
LOG_FILTER (0 or 1)	Specifies whether or not the list of active filters enclosed by single quotes are written into enqueue records in the log file just prior to the diagnostics field. The default is 0 (do not write lists into enqueue records).

Table 4-16 Option File Options *(Continued)*

Options	Description
LOG_FORMAT (1, 2, or 3)	Controls formatting options for the mail.log file. A value of 1 (the default) is the standard format. A value of 2 requests non-null formatting: empty address fields are converted to the string "<>." A value of 3 requests counted formatting: all variable length fields are preceded by N, where N is a count of the number of characters in the field.
LOG_FRUSTRATION_LIMIT	Specifies the limit of "frustration counts." In a process, if repeated retries of writing a counter fails, the "frustration count" is incremented. Once the count reaches this limit, that process stops attempting to write counters.
LOG_HEADER (0 or 1)	Controls whether the MTA writes message headers to the mail.log file. A value of 1 enables message header logging. The specific headers written to the log file are controlled by a site-supplied log_header.opt file. The format of this file is that of other MTA header option files. For example, a log_header.opt file containing the following would result in writing the first To and the first From header per message to the log file. A value of 0 (the default) disables message header logging: To: MAXIMUM=1 From: MAXIMUM=1 Defaults: MAXIMUM=-1
LOG_LOCAL (0 or 1)	Controls whether the domain name for the local host is appended to logged addresses that don't already contain a domain name. A value of 1 enables this feature, which is useful when logs from multiple systems running the MTA are concatenated and processed. A value of 0, the default, disables this feature.
LOG_MESSAGE_ID (0 or 1)	Controls whether message IDs are saved in the mail.log file. A value of 1 enables message ID logging. A value of 0 (the default) disables it.
LOG_MESSAGES_SYSLOG (0 or 1)	Sends MTA message log file entries to syslog (UNIX) or event log (Windows NT). 0 is the default and indicates that syslog (event log) logging is not performed. A value of 1 indicates that syslog logging is performed.
LOG_PROCESS (0 or 1)	Includes the enqueuing process ID in the MTA's log entries.
LOG_SNDOPR (0 or 1)	Controls the production of syslog messages by the MTA message logging facility.
LOG_SIZE_BINS	Specifies the bin sizes for message size range counters. The value is a comma-separated list of up to five integers. The default values are 2, 10, 50, 100, 500.

Table 4-16 Option File Options *(Continued)*

Options	Description
LOG_USERNAME (0 or 1)	The LOG_USERNAME option controls whether or not the username associated with a process that enqueues mail is saved in the `mail.log` file. Note that messages submitted via SMTP with authentication (SMTP AUTH) will be considered to be owned by the username that authenticated, prefixed with the asterisk, *, character. A value of 1 enables username logging. When username logging is enabled, the username will be logged after the final form envelope To: address field in log entries---and after the message ID, if LOG_MESSAGE_ID=1 is also enabled. A value of 0 (the default) disables username logging.
MAIL_OFF (String)	Specifies the comment string that disables mail delivery for list addresses. The default is NOMAIL.
MAP_NAMES_SIZE (Integer > 0)	Specifies the size of the mapping table name table, and thus the total number of mapping table cannot exceed this number. The default is 32.
MAX_ALIAS_LEVELS (Integer)	Controls the degree of indirection allowed in aliases; that is, how deeply aliases may be nested, with one alias referring to another alias, and so forth. The default value is 10.
MAX_FILEINTOS (Integer)	Specifies the maximum number of files that may be specified by a mailbox filter's fileinto operator. Default: 10
MAX_FORWARDS (Integer)	Specifies the maximum number of forwarding addresses that may be specified by a mailbox filter's forward operator. Default: 10
MAX_HEADER_BLOCK_USE (Real Number Between 0 and 1)	Controls what fraction of the available message blocks can be used by message headers.
MAX_HEADER_LINE_USE (Real Number Between 0 and 1)	Controls what fraction of the available message lines can be used by message headers.
MAX_INTERNAL_BLOCKS (Integer)	Specifies how large (in MTA blocks) a message the MTA keeps entirely in memory; messages larger than this size is written to temporary files. The default is 30. For systems with lots of memory, increasing this value may provide a performance improvement.

Table 4-16 Option File Options *(Continued)*

Options	Description
MAX_LOCAL_RECEIVED_LINES (Integer)	As the MTA processes a message, it scans any Received: header lines attached to the message looking for references to the official local host name. (Any Received line that the MTA inserts contains this name.) If the number of Received lines containing this name exceeds the MAX_LOCAL_RECEIVED_LINES value, the message is entered in the MTA queue in a held state. The default for this value is 10 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue.
MAX_MIME_LEVELS (Integer)	Specify the maximum depth to which the MTA should process MIME messages. The default is 100, which means that the MTA processes up to 100 levels of message nesting.
MAX_MIME_PARTS (Integer)	Specify the maximum number of MIME parts that the MTA should process in a MIME message.
MAX_MR_RECEIVED_LINES (Integer)	As the MTA processes a message, it counts the number of MR_Received: header lines in the message's header. If the number of MR-Received: lines exceeds the MAX_MR_RECEIVED_LINES value, the message is entered into the MTA queue in a held state. The default value for this option is 20 if no value is specified in the Option File. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue.
MAX_RECEIVED_LINES (Integer)	As the MTA processes a message, it counts the number of Received: header lines in the message's header. If the number of Received lines exceeds the MAX_RECEIVED_LINES value, the message is entered in the MTA queue in a held state. The default for this value is 50 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue.
MISSING_RECIPIENT_POLICY (Integer)	Legalizes messages that lack any recipient headers.
MAX_SIEVE_LIST_SIZE (Integer)	Controls the number of strings that can appear in a list construct in MTA sieve scripts. The default is 64.
MAX_TOTAL_RECEIVED_LINES (Integer)	As the MTA processes a message, it counts the number of Received:, MR-Received:, and X400-Received: header lines in the message's header. If the number of all such header lines exceeds the MAX_TOTAL_RECEIVED_LINES value, the message is entered into the MTA queue in a held state. The default value is 100 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue.
MAX_URLS (Integer)	Specifies the URL to query for address reversal. Standard LDAP URL syntax is used, except omitting the LDAP server and port which are instead specified via the LDAP_HOST and LDAP_PORT options.

Table 4-16 Option File Options *(Continued)*

Options	Description
MAX_X400_RECEIVED_LINES (Integer)	As the MTA processes a message, it counts the number of X400-Received: header lines in the message's header. If the number of Received: lines exceeds the MAX_X400_RECEIVED_LINES value, the message is entered into the MTA queue in a held state. The default value for this option is 50 if no value is specified in the Option File. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue.
NORMAL_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size is downgraded to non-urgent priority. This priority, in turn, affects the processing priority of the message—how quickly the Job Controller processes the message.
NON_URGENT_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: Messages above the specified size is downgraded to lower than nonurgent priority. The value is interpreted in terms of MTA blocks, as specified by the BLOCK_SIZE option. Note also that the nonurgentblocklimit channel keyword may be used to impose such downgrade thresholds on a per channel basis.
NOTARY_DECODE_FLAGS (-1, 0, or 1)	Specifies the decoding condition of encoded words. If set to 1, NOTARY_DECODE_FLAGS causes the subset of the original message headers that are added tot he first part of a DSN by the %H substitution to be decoded and converted to match the charset of the first part. A value of 0 decodes the subset of encoded words in the header that matches the charset of the first part; no charset conversion is performed. A value of -1 disables decoding of encoded words unconditionally. Caution should be used with a setting of 1, as information loss can occur and confusion can result when a rich charset like UTF-8 is converted to a limited charset like ISO-8859-1 or US-ASCII. The default is 0.

Table 4-16 Option File Options *(Continued)*

Options	Description
OPTIN_USER_CARRYOVER (Integer)	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 interpretation is as follows: Bit-0 Value-1: When set, each LDAP user entry overrides any previously active user/domain optins unconditionally. Bit-1 Value-2: When set, it overrides any previous user, domain, or alias optins that were active if a user's domain has an optin attribute. Bit-2 Value-4: When set, it overrides any previous user, domain, or alias optins that were active, if a user has an optin attribute. Bit-3 Value-8: When set, an optin specified by an (optin) nonpositional parameter overrides any previous user, domain, or alias optins that were active. The value of this option defaults to 0. Optins will accumulate if one user has a delivery option that forwards to another user. The default ensures that site security policies will be effective when forwarding mail.
OR_CLAUSES (0 or 1)	Specifies mailing list access controls are OR'ed by default, instead of AND'ed.
POST_DEBUG (0 or 1)	Specifies whether or not debugging output is produced by the MTA's periodic delivery job. If enabled with a value of 1, this output is produced in the post.log file. The default value of 0 disables this output.
RECEIVED_DOMAIN (String)	Sets the domain name to use when constructing Received headers. By default, the official host name of the local channel is used.
RECEIVED_VERSION (String)	Sets the Sun Java System Messaging Server version string that is to be used when constructing Received: header lines. By default, the string "(Sun Java System Messaging Server *version-info*)" is used; use of the default is strongly recommended. Note that this option is a complement to the (also not recommended) CUSTOM_VERSION_STRING TCP/IP SMTP channel option. In the above description, note the mention of *constructing* a Received: header line; that is, this option does not change already present Received: header lines, but rather only affects what is used when generating a new Received: header line. Also note that this option is option and the CUSTOM_VERSION_STRING option should not be used. A non-ASCII string could be specified, but the MTA would then have to MIME encode the non-ASCII characters. Since user agent handling of MIME encoded header lines is not always useful, specifying a non-ASCII value would be inadvisable. So while the value is not strictly limited to being an ASCII string, it is not recommended to use anything other than ASCII.

Table 4-16 Option File Options *(Continued)*

Options	Description
RETURN_ADDRESS (String)	Sets the return address for the local postmaster. The local postmaster's address is postmaster@*localhost* by default, but it can be overridden with the address of your choice. Care should be taken in the selection of this address—an illegal selection may cause rapid message looping and pileups of huge numbers of spurious error messages.
RETURN_DEBUG (0 or 1)	Enables or disables debugging output in the nightly message bouncer batch job. A value of 0 disables this output (the default), while a value of 1 enables it. Debugging output, if enabled, appears in the output log file, if such a log file is present. The presence of an output log file is controlled by the crontab entry for the return job.
RETURN_DELIVERY_HISTORY (0 or 1)	Controls whether or not a history of delivery attempts is included in returned messages. The delivery history provides some indication of how many delivery attempts were made and, in some cases, indicates the reason the delivery attempts failed. A value of 1 enables the inclusion of this information and is the default. A value of 0 disables return of delivery history information. The HISTORY_TO_RETURN option controls how much history information is actually returned.
RETURN_ENVELOPE (Integer)	Takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether 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 addresses. Note that the use of blank address is mandated by RFC 1123. However, some systems do not handle blank-envelope-from-address properly and may require the use of this option. Bit 1 (value = 2) controls whether the MTA replaces all blank envelope addresses with the address of the local postmaster. Again, this is used to accommodate noncompliant systems that don't conform to RFC 821, RFC 822, or RFC 1123. Note that the returnenvelope channel keyword can be used to impose this sort of control on a per-channel basis.
RETURN_PERSONAL (String)	Specifies the personal name to use when the MTA generates postmaster messages (for example, bounce messages). By default, the MTA uses the string, Internet Mail Delivery.
RETURN_UNITS (0 or 1)	Controls the time units used by the message return system. A value of 0 selects units of days. A value of 1 selects units of hours. By default, units of days are used. Return job scheduling is controlled by the local.schedule.return_job configutil parameter.
REVERSE_ADDRESS_CACHE_SIZE	Specifies the maximum size of the address reversal cache. Default: 100000
REVERSE_ADDRESS_CACHE_TIMEOUT	Specifies the timeout, in seconds, for entries in the address reversal cache. Default: 600

Table 4-16 Option File Options *(Continued)*

Options	Description
REVERSE_ENVELOPE (0 or 1)	Controls whether the MTA applies the address reversal to envelope From addresses as well as header addresses. This option has no effect if the USE_REVERSE_DATABASE option is set to 0 or if the reverse database and reverse mapping does not exist. The default is 1, which means that the MTA attempts to apply the database to envelope From addresses. A value of 0 disables this use of the address reversal database.
REVERSE_URL (URL)	Specifies the URL to query for address reversal. Standard LDAP URL syntax is used, except omitting the LDAP server and port, which are instead specified using the LDAP_HOST and LDAP_PORT options.
ROUTE_TO_ROUTING_HOST (0 or 1)	Specifies (value set to 1) that the Messaging Server routes all addresses associated with the domain to the first host listed in the mailRoutingHosts attribute. A value of 0 indicates that a failure to match an existant mailRoutingHosts attribute against causes the domain to be treated as non-local; addresses are routed onward according to their rewrite rules. The default is 0.
SEPARATE_CONNECTION_LOG (0 or 1)	Controls whether the connection log information generated by setting LOG_CONNECTION =1 is stored in the usual the MTA message logging files, mail.log* or is stored separately in connection.log* files. The default (0) causes connection logging to be stored in the regular message log files; 1 causes the connection logging to be stored separately.
SIEVE_USER_CARRYOVER (0 or 1)	Controls how user sieves are combined when forwarding occurs.This is a bit-encoded value; the interpretation is as follows: Bit-0 Value-1: When set, user-to-user forwarding cancels the domain and user scripts associated with the original user entry. The value of this option defaults to 0. Using this option is not recommended, because it prevents a user from filtering mail prior to it being forwarded.
SNDOPR_PRIORITY (Integer)	Sets the syslog level of syslog messages or the severity of the Windows NT event log entry. For syslog, this option corresponds to the priority argument of the syslog call. Both the facility and severity can be set by applying a logical OR operation to the desired values. On Solaris, see /usr/include/sys/syslog.h for a definition of valid values. Be sure to coordinate setting the SNDOPR_PRIORITY option with how syslog messages are handled, as controlled by the syslog.conf file. The default for UNIX is 5; the default for Windows NT is 1.
STRICT_REQUIRE (0 or 1)	Enforces strict Sieve compliance for location of require clauses. The default is 0.

Table 4-16 Option File Options *(Continued)*

Options	Description
STRING_POOL_SIZE (Integer <= 10,000,000)	Controls the number of character slots allocated to the string pool used to hold rewrite rule templates and alias list members. A fatal error occurs if the total number of characters consumed by these parts of the configuration and alias files exceeds this limit. The default is 60,000; the maximum allowed value is 10,000,000.
URGENT_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size are downgraded to normal priority. This priority, in turn, affects the Job Controller's processing priority for processing the message. The value is interpreted in terms of the MTA blocks, as specified by the BLOCK_SIZE option. Note also that the urgentblocklimit channel keyword may be used to impose such downgrade thresholds on a per-channel basis.
USE_ALIAS_DATABASE (0 or 1)	Controls whether the MTA uses the alias database as a source of system aliases for local addresses. The default (1), means that the MTA checks the database if it exists. A value of 0 disables this use of the alias database.
USE_DOMAIN_DATABASE (0 or 1)	Controls the use of the domain database. The default (1) means that the MTA checks the database if it exists.
USE_FORWARD_DATABASE (Integer)	Control use of the forward database.
USE_ORIG_RETURN	Controls the bit encoded field.
USE_PERMANENT_ERRORS (0-15)	Controls whether or not certain errors returned by the MTA are marked as temporary or permanent. Each bit in this option corresponds to a specific error condition. When set, this option instructs the MTA to return a permanent error. The value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in Table 4-18. The default value is 0.
USE_PERSONAL_ALIASES (0 or 1)	Controls whether or not the MTA makes use of personal alias databases as a source of aliases for local addresses. The default is 1, which means that the MTA checks such databases, if they exist. A value of 0 disables personal aliases and makes them unavailable to all users.
USE_REVERSE_DATABASE (0-31)	Controls whether the MTA uses the address reversal database and REVERSE mapping as a source of substitution addresses. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in Table 4-19. Note that bit 0 is the least significant bit. The default value for USE_REVERSE_DATABASE is 5, which means that the MTA reverse envelope From addresses and both backward and forward pointing addresses after they have passed through the normal address rewriting process. Simple address strings are presented to both REVERSE mapping and the reverse database. A value of 0 disables the use of the address reversal completely.

Table 4-16 Option File Options *(Continued)*

Options	Description
`WILD_POOL_SIZE (integer)`	Controls the total number of patterns that appear throughout mapping tables. the default is 8000. The maximum allowed is 200,000.

Table 4-17 `DOMAIN_UPLEVEL` Bit Values

Value	Bit 1 (Address Lookup)	Bit 0 (Domain Lookup)
0	0 - Look for address only in the given domain. Do not look in other domains	0 - Look only for given domain.
1	0 - Look for address only in the given domain. Do not look in other domains.	1 - Look for given domain and subtrees of given domain.
2	1 - Lookup address in the given domain, in subdomains, and in domains aliased to given domain or subdomains.	0 - Look only for given domain.
3	1 - Lookup address in the given domain, in subdomains, and in domains aliased to given domain or subdomains.	1 - Look for given domain and subtrees of given domain.

Table 4-18 `USE_PERMANENT_ERRORS` Bit Values

Bit	Value	Error
0	1	Mailbox is temporarily disabled (inactive).
1	2	Group is temporarily disabled (inactive).
2	4	User is over quota; cannot receive new mail.
3	8	Various alias expansion errors.

Table 4-19 `USE_REVERSE_DATABASE` Bit Values

Bit	Value	Usage
0	1	When set, address reversal is applied to addresses after they have been rewritten by the MTA address rewriting process.
1	2	When set, address reversal is applied before addresses have had the MTA address rewriting applied to them.

Table 4-19 USE_REVERSE_DATABASE Bit Values *(Continued)*

Bit	Value	Usage
2	4	When set, address reversal is applied to all addresses, not just to backward pointing addresses.
3	8	When set, channel-level granularity is used with REVERSE mapping. REVERSE mapping table (pattern) entries must have the form (note the vertical bars [|]). `source-channel|destination-channel|address`
4	16	When set, channel-level granularity is used with address reversal database entries. Reversal database entries must have the form (note the vertical bars [|]). `source-channel|destination-channel|address`

Table 4-20 LDAP_USE_ASYNC Bit Values

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

Header Option Files

Some special option files may be associated with a channel that describe how to trim the headers on messages queued to that channel or received by that channel. This facility is completely general and may be applied to any channel; it is controlled by the `headertrim`, `noheadertrim`, `headerread`, and `noheaderread` channel keywords.

Various MTA channels have their own channel-level option files as well. Header option files have a different format than other MTA option files, so a header option file is always a separate file.

Header Option File Location

For destination channel based header trimming to be applied upon message *enqueue* after normal header processing, the MTA looks in the `config` directory (*msg_svr_base*`/config`) for header options files with names of the form *channel*`_headers.opt`, where *channel* is the name of the channel with which the header option file is associated. The `headertrim` keyword must be specified on the channel to enable the use of such a header option file.

For source channel based header trimming to be applied upon message *enqueue* before normal header processing, the MTA looks in the `config` directory (*msg_svr_base*`/config`) for header options files with names of the form *channel*`_read_headers.opt`, where *channel* is the name of the channel with which the header option file is associated. The `headerread` keyword must be specified on the channel to enable the use of such a header option file.

Header option files should be world readable.

Header Option File Format

Simply put, the contents of a header option file are formatted as a set of message header lines. Note, however, that the bodies of the header lines do not conform to RFC 822.

The general structure of a line from a header options file is:

Header-name: OPTION=VALUE, OPTION=VALUE, OPTION=VALUE, ...

Header-name is the name of a header line that the MTA recognizes (any of the header lines described in this manual may be specified, plus any of the header lines standardized in RFC 822, RFC 987, RFC 1049, RFC 1421, RFC 1422, RFC 1423, RFC 1424, RFC 1327, and RFC 1521 (MIME).

Header lines not recognized by the MTA are controlled by the special header line name Other:. A set of options to be applied to all header lines not named in the header option file can also be given on a special Defaults: line. The use of Defaults: guards against the inevitable expansion of the MTA's known header line table in future releases.

Various options can then be specified to control the retention of the corresponding header lines. The available options are listed in Table 4-21.

Table 4-21 Header options

Option	Description
ADD (Quoted String)	Creates a new header line of the given type. The new header line contains the specified string. The header line created by ADD appears after any existing header lines of the same type. The ADD option cannot be used in conjunction with the Defaults header line type; it is ignored if it is specified as part of an Other: option list.
FILL (Quoted String)	Creates a new header line of the given type only if there are no existing header lines of the same type. The new header line contains the specified string. The FILL option cannot be used in conjunction with the header line type; it is ignored if it is specified as part of an Other option list.
GROUP (Integer 0 or 1)	Controls grouping of header lines of the same type at a particular precedence level. A GROUP value of 0 is the default, and indicates that all header lines of a particular type should appear together. A value of 1 indicates that only one header line of the respective type should be output and the scan over all header lines at the associated level should resume, leaving any header lines of the same type unprocessed. Once the scan is complete it is then repeated in order to pick up any remaining header lines. This header option is primarily intended to accommodate Privacy Enhanced Mail (PEM) header processing.
LINELENGTH (Integer)	Controls the length at which to fold headers. See the headerlinelength channel keyword.
MAXCHARS (Integer)	Controls the maximum number of characters that can appear in a single header line of the specified type. Any header line exceeding that length is truncated to a length of MAXCHARS. This option pays no attention to the syntax of the header line and should never be applied to header lines containing addresses and other sorts of structured information. The length of structured header lines should instead be controlled with the maxheaderchars and maxheaderaddrs channel keywords.
MAXIMUM (Integer)	Controls the maximum number of header lines of this type that may appear. This has no effect on the number of lines; after wrapping, each individual header line can consume. A value of -1 is interpreted as a request to suppress this header line type completely.

Table 4-21 Header options *(Continued)*

Option	Description
MAXLINES (Integer)	Controls the maximum number of lines all header lines of a given type may occupy. It complements the MAXIMUM option in that it pays no attention to how many header lines are involved, only to how many lines of text they collectively occupy. As with the MAXIMUM option, headers are trimmed from the bottom to meet the specified requirement.
PRECEDENCE (Integer)	Controls the order in which header lines are output. All header lines have a default precedence of zero. The smaller the value, the higher the precedence. Positive PRECEDENCE values push header lines toward the bottom of the header while negative values push them toward the top. Equal precedence ties are broken using the MTA's internal rules for header line output ordering.

RELABEL (header name)	Changes a header line to another header line; that is, the name of the header is changed, but the value remains the same. For instance,
	`X-MSMail-Priority: RELABEL="Priority"`
	`X-Priority: RELABEL="Importance"`

Tailor File

The MTA tailor file (`imta_tailor`) is an option file in which the location of various MTA components are set. This file must always exist in the *msg_svr_base*`/config` directory for the MTA to function properly. The file may be edited to reflect the changes in a particular installation. Some options in the file should not be edited. The MTA should be restarted after making any changes to the file. It is preferable to make the changes while the MTA is down.

An option setting has the form:

```
option=value
```

The value can be either a string or an integer, depending on the option's requirements. Comments are allowed. Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored. Options that are available and can be edited are shown in Table 4-22.

Table 4-22 tailor File Options

Option	Description
IMTA_ALIAS_DATABASE	The alias database. The default is *msg_svr_base*/data/db/aliasesdb.
IMTA_ALIAS_FILE	The MTA aliases file. Aliases not set in the directory, for example, postmaster, are set in this file. The default is *msg_svr_base*/config/aliases.
IMTA_CHARSET_DATA	Specifies where the MTA compiled character set data is located. The default is *msg_svr_base*/config/charset_data.
IMTA_CHARSET_OPTION_FILE	File used for charset conversion options. The default is *msg_svr_base*/config/option_charset.dat.
IMTA_COM	Specifies where the MTA command definition files are located. The default is *msg_svr_base*/bin/msg/imta/bin/.
IMTA_CONFIG_DATA	Compiled configuration for the MTA. The default is *msg_svr_base*/imta/lib/config_data.
IMTA_CONFIG_FILE	The MTA configuration file. Rewrite rules and per-channel options are set in this file. The default is *msg_svr_base*/config/imta.cnf.
IMTA_CONVERSION_FILE	File to set rules for the conversion channel. The default is *msg_svr_base*/config/conversions.
IMTA_DISPATCHER_CONFIG	The MTA dispatcher's configuration file. The default is *msg_svr_base*/config/dispatcher.cnf.
IMTA_DOMAIN_DATABASE	Database used to store additional rewrite rules. The default is *msg_svr_base*/data/db/domaindb
IMTA_DNSRULES	The MTA DNS configuration library. The default is *msg_svr_base*/imta/lib/imdnsrules.so.
IMTA_EXE	Location of the MTA executables. The default is *msg_svr_base*/bin/msg/imta/bin.
IMTA_FORWARD_DATABASE	Not used.
IMTA_GENERAL_DATABASE	Provided for each site's customized usage. Generally, lookups can be embedded in mappings and rewrite rules. The default is *msg_svr_base*/config/generaldb.
IMTA_HELP	Location of the help files for the MTA utility. The default is *msg_svr_base*/imta/lib.
IMTA_JBC_CONFIG_FILE	The MTA Job Controller's configuration file. The default is *msg_svr_base*/config/job_controller.cnf.

Table 4-22 tailor File Options *(Continued)*

Option	Description
IMTA_LANG	Locale of the MTA's notary messages. By default it is *msg_svr_base*/imta/locale/C/LC_MESSAGES.
IMTA_LIB	Directory where the MTA libraries and executables are stored. The default is *msg_svr_base*/imta/lib/.
IMTA_LIBUTIL	The MTA utility library. By default it is *msg_svr_base*/lib/libimtautil.so.1.
IMTA_LOG	Location of the MTA log files. The default is *msg_svr_base*/imta/log.
IMTA_MAPPING_FILE	File used for setting access control rules, reverse mapping rules, forward mapping rules, and so forth. The default value is *msg_svr_base*/config/mappings.
IMTA_NAME_CONTENT_FILE	Location of file used by the MTA for certain attachment handling labeling. The default is *msg_svr_base*/config/name_content.dat.
IMTA_OPTION_FILE	Name of the MTA's option file. The default is *msg_svr_base*/config/option.dat.
IMTA_QUEUE	The MTA message queue directory. The default is *msg_svr_base*/imta/queue. **CAUTION**: Do not add any files or directories in the MTA queue directory as this causes 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.
IMTA_RETURN_PERIOD	Controls the return of expired messages and the generation of warnings. The default value for this option is 1. If this options is set to an integer value N, then the associated action is only performed every N times the return job runs. By default, the return job runs once every day.
IMTA_RETURN_SPLIT_PERIOD	Controls splitting of the mail.log file. The default value for this option is 1. If this options is set to an integer value *N*, then the associated action is only performed every *N* times the return job runs. By default, the return job runs once every day.
IMTA_REVERSE_DATABASE	The MTA reverse database. This database is used for rewriting From addresses. The default is *msg_svr_base*/data/db/reversedb.
IMTA_ROOT	Base directory for the MTA installation. The default is *msg_svr_base*/imta/.
IMTA_SYSTEM_FILTER_FILE	Specifies the location of the MTA system filter file. The value of this option can be either a file name or a URL.
IMTA_TABLE	The MTA configuration directory. The default is *msg_svr_base*/config/.
IMTA_USER	Name of the postmaster. The default is inetmail. If this is changed be sure to edit the *msg_svr_base*/config/aliases file to reflect the change to the postmaster address.

Table 4-22 tailor File Options *(Continued)*

Option	Description
IMTA_USER_PROFILE_DATABASE	Database used for storing user's vacation, forwarding, and program delivery information. The default is *msg_svr_base*/data/db/profiledb.
IMTA_USER_USERNAME	Specifies the userid of the subsidiary account the MTA uses for certain "non-privileged" operations—operations which it doesn't want to perform under the usual MTA account. The default is nobody.
IMTA_VERSION_LIMIT	Maximum versions of log files to be preserved while purging old log files. The default value is 5.
IMTA_WORLD_GROUP	Can perform certain privileged operations as a member of this group. The default is mail.

Job Controller Configuration

At startup, the Job Controller reads a configuration file that specifies parameters, pools, and channel processing information. This configuration information is specified in the file job_controller.cnf in the *msg_svr_base*/config/ directory.

For more information on the Job Controller, see the "About MTA Services and Configuration" chapter in the *Sun Java System Messaging Server Administration Guide.*

Job Controller Configuration File

In accordance with the format of the MTA option files, the Job Controller configuration file contains lines of the form:

```
option=value
```

In addition to option settings, the file may contain a line consisting of a section and value enclosed in square-brackets ([ ]) in the form:

```
[section-type=value]
```

Such a line indicates that option settings following this line apply only to the section named by *value*. Initial option settings that appear before any such section tags apply globally to all sections. Per section option settings override global defaults for that section. Recognized section types for the Job Controller configuration file are `POOL`, to define pools and their parameters, and `CHANNEL`, to define channel processing information and `PERIODIC_JOB` for the various periodic jobs started by the Job Controller. The `PERIODIC_JOB` is deprecated and will be removed in a future release. Use the `local.schedule.periodic_job` configutil parameter instead.

Any options permitted on `POOL` or `CHANNEL` sections can be specified at the beginning (general options), thus becoming the default for the option.

The Job Controller configuration file options are described in the following three tables (Table 4-23, Table 4-24, and Table 4-25). They are split into general options, pool options, and channel options groups.

Table 4-23 shows the general Job Controller configuration options.

Table 4-23 General Job Controller Configuration File Options

Option	Description
`COMMAND`	Specifies the command to be run periodically in a `PERIODIC_JOB` section. The `PERIODIC_JOB` is deprecated and will be removed in a future release. Use the `local.schedule.periodic_job` configutil parameter instead.
`DEBUG=`*integer*	If `DEBUG` is set to a value other than zero, the MTA writes debugging information to a file in the *msg_svr_base*`/imta/log` directory named `job_controller-`*uniqueid*, where *uniqueid* is a unique ID string that distinctively identifies the file name. The `imsimta purge` utility recognizes the *uniqueids* and can be used to remove older log files. The value for `DEBUG` is a bit mask specifying what sort of debugging information is requested: • 1—Trace protocol messages between the Job Controller and other MTA components. • 2—More detailed analysis of the messages and interactions. • 4—State change events. • 8—Trace rebuild decisions. • 16—Dump each queue on every queue action. • 32—Be cautious about deleting items from queues. • 64—Perform queue integrity check on every queue operation • 128—Verbose output about operation of select. Specifying bit 16 can cause log files to grow very quickly. Specifying 32 does not generate any more output, and should only be used in extreme cases. If `DEBUG` is not specified, it defaults to 0.

Table 4-23 General Job Controller Configuration File Options *(Continued)*

Option	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 any new message files that are missing from the Job Controller's list of messages that need to be added. 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 4-23 General Job Controller Configuration File Options *(Continued)*

Option	Description
TIME=*time_spec*	Specifies the time and frequency that a periodic job is run in a PERIODIC_JOB section. By default, this is /4:00, which means every four hours. The format of *time_spec* is *HH*:*MM*/*hh*:*mm* or /*hh*:*mm*. *hh*.*mm* is the interval between the events in hours (*h*) and minutes (*m*). *HH*:*MM* is the first time in a day that a job should occur. For example, specifying 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then. The PERIODIC_JOB is deprecated and will be removed in a future release. Use the local.schedule.periodic_job configutil parameter instead.

Table 4-24 describes the POOL option for the Job Controller configuration.

Table 4-24 Job Controller POOL Option

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.

Table 4-25 describes the CHANNEL options for the Job Controller configuration.

Table 4-25 Job Controller CHANNEL Options

Option	Description
MASTER_COMMAND=*file_spec*	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller to run the channel and dequeue messages outbound on that channel. If set outside of a section, it is used as the default by any [CHANNEL] section that doesn't specify a MASTER_COMMAND. This option is ignored inside of a [POOL] section.
MAX_LIFE_AGE=*integer*	Specifies the maximum life time for a channel master job in seconds. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 1800 (30 minutes) is used.
MAX_LIFE_CONNS=*integer*	In addition to the maximum life age parameter, the life expectancy of a channel master job is limited by the number of times it can ask the Job Controller if there are any messages. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 300 is used.

Table 4-25 Job Controller CHANNEL Options

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

Dispatcher

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

Dispatcher Configuration File

The Dispatcher configuration information is specified in the *msg_svr_base*/imta/dispatcher.cnf file. 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.

Configuration File Format

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.

Table 4-26 shows the available options.

Table 4-26 Dispatcher configuration file options

Option	Description
BACKLOG=*integer*	Controls the depth of the TCP backlog queue for the socket. The default value for each service is MAX_CONNS*MAX_PROCS (with a minimum value of 5). This option should not be set higher than the underlying TCP/IP kernel supports.
DEBUG	Enables debugging output. Enabling all debugging is done by setting the option to -1. The actual meaning of each bit is described in Table 4-27.

DNS_VERIFY_DOMAIN	Specifies the host name or IP address of source against which to check incoming connections. Various groups maintain information about unsolicited email sources or open relay sites. Some sites check incoming IP connections against the lists maintained by such groups. Up to five DNS_VERIFY_DOMAIN options can be specified for each service. Note that SMTP is typically the only service for which such checks make sense. For example:
	`[SERVICE=SMTP]`
	`PORT=25`
	`DNS_VERIFY_DOMAIN=rbl.maps.siroe.com`
	`DNS_VERIFY_DOMAIN=dul.maps.siroe.com`
	If this options is enabled on a well known port (25, 110, or 143), then a standard message such as the one below is sent before the connection is closed:
	`500 5.7.1 access_control: host 192.168.51.32 found on DNS list and rejected`
	If you wish the MTA to log such rejections, the 24th bit of the Dispatcher debugging DEBUG option can be set (DEBUG=16%1000000) to cause logging of the rejections to the `dispatcher.log` file. Log entries take the following form:
	`access_control: host a.b.c.d found on DNS list and rejected`

Table 4-26 Dispatcher configuration file options *(Continued)*

Option	Description

ENABLE_RBL=0 or 1	Specifying ENABLE_RBL=1 causes the Dispatcher to compare incoming connections to the "Black Hole" list at maps.siroe.com. For instance, if the Dispatcher receives a connection from 192.168.51.32, then it attempts to obtain the IP address for the hostname 32.51.168.192.rbl.maps.siroe.com. If the query is successful, the connection is closed rather than handed off to a worker process. If this option is enabled on a well-known port (25, 110, or 143), then a standard message such as the one below is sent before the connection is closed:
	`5.7.1 Mail from 192.168.51.32 refused, see http://maps.siroe.com/rbl/`
	If you want the MTA to log such rejections, set bit 24 of the Dispatcher debugging DEBUG option, DEBUG=16%1000000, to cause logging of the rejections to the dispatcher.log file; entries take the form:
	`access_control: host a.b.c.d found on DNS list and rejected`
	See the section "To Use DNS Lookups Including RBL Checking for SMTP Relay Blocking" in the "Mail Filtering and Access Control" chapter of the *Messaging Server Administrator's Guide* for more information.

HISTORICAL_TIME=*integer*	Controls how long the expired connections (those that have been closed) and processes (those that have exited) remain listed for statistical purpose in the Dispatcher statistics.
INTERFACE_ADDRESS=*IP address*	The INTERFACE_ADDRESS option can be used to specify the IP address interface to which the Dispatcher service should bind. By default, the Dispatcher binds to all IP addresses. But for systems having multiple network interfaces each with its own IP address, it may be useful to bind different services to the different interfaces. Note that if INTERFACE_ADDRESS is specified for a service, then that is the only interface IP address to which that Dispatcher service bind. Only one such explicit interface IP address may be specified for a particular service (though other similar Dispatcher services may be defined for other interface IP addresses).
IDENT=*0 or 1*	If IDENT=1 is set for a service, it causes the Dispatcher to try an IDENT query on incoming connections for that service, and to note the remote username (if available) as part of the Dispatcher statistics. The default is IDENT=0, meaning that no such query is made.
IMAGE=*file specification*	Specifies the image that is run by server processes when created by the Dispatcher. The specified image should be one designed to be controlled by the Dispatcher.
LOGFILE=*file specification*	Causes the Dispatcher to direct output for corresponding server processes to the specified file. LOGFILE can include a %s which includes the local system's hostname in the file specification. For example, LOGFILE=tcp_smtp_server_%s.log on node freddy results in log files with the name tcp_smtp_server_freddy.log-*.

Table 4-26 Dispatcher configuration file options *(Continued)*

Option	Description
MAX_CONNS=*integer*	Specifies a maximum number of connections that may be active on any server process. The MAX_CONNS option affects the Dispatcher's management of connections. The default value for MAX_CONNS is 10. The maximum possible value for MAX_CONNS is 50. The choice of setting this option is mainly a performance issue relating to the number of processes and the size of the process virtual address space. Setting MAX_CONNS to higher values allows more connections, but at the potential cost of decreased performance for each individual connection. If it is set to 1, then for every incoming client connection, only one server process is used. Note that the value of MAX_CONNS multiplied by the value of MAX_PROCS controls the maximum number of simultaneous connections that can be accepted.
MAX_HANDOFFS=*integer*	Specifies the maximum number of concurrent asynchronous hand-offs in progress that the Dispatcher allows for newly established TCP/IP connections to a service port. The default value is 5.
MAX_IDLE_TIME=*integer*	Specifies the maximum idle time for a server process. When an server process has had no active connections for this period, it becomes eligible for shutdown. This option is only effective if there are more than the value of MIN_PROCS server processes currently in the Dispatcher's pool for this service.
MAX_LIFE_CONNS	Specifies the maximum number of connections an server process can handle in its lifetime. Its purpose is to perform worker-process housekeeping.
MAX_LIFE_TIME=*integer*	Requests that server processes be kept only for the specified number of seconds. This is part of the Dispatcher's ability to perform worker-process housekeeping. When an server process is created, a countdown timer is set to the specified number of seconds. When the countdown time has expired, the SMTP server process is subject to shutdown. Default value is 86400 (one day).
MAX_PROCS=*integer*	Controls the maximum number of server processes that are created for this service.
MAX_SHUTDOWN=*integer*	Specifies the maximum number of server processes which can be in the shutdown state. In order to provide a minimum availability for the service, the Dispatcher does not shut down server processes that might otherwise be eligible for shutdown if shutting them down results in having more than MAX_SHUTDOWN server processes for the service in the shutdown statue. This means that processes that are eligible for shutdown can continue running until a shutdown "slot" is available.
MIN_CONNS=*integer*	Determines the minimum number of connections that each Worker Process must have before considering the addition of a new server process to the pool of currently available server processes. The Dispatcher attempts to distribute connections evenly across this pool.
MIN_PROCS=*integer*	Determines the minimum number of server processes that are created by the Dispatcher for the current service. Upon initialization, the Dispatcher creates this many detached processes to start its pool. When a process is shut down, the Dispatcher ensures that there are at least this many available processes in the pool for this service.

Table 4-26 Dispatcher configuration file options *(Continued)*

Option	Description
PARAMETER	The interpretation and allowed values for the PARAMETER option are service specific. In the case of an SMTP service, the PARAMETER option may be set to CHANNEL=channelname, to associate a default TCP/IP channel with the port for that service. For instance: [SERVICE=SMTP_SUBMIT] PORT=587 ... PARAMETER=CHANNEL=tcp_incoming This can be useful if you want to run servers on multiple ports—if your internal POP and IMAP clients have been configured to use a port other than the normal port 25 for message submission, separating their message traffic from incoming SMTP messages from external hosts—and if you want to associate different TCP/IP channels with the different port numbers.
PORT=*integer*...	Specifies the TCP port(s) to which the Dispatcher listens for incoming connections for the current service. Connections made to this port are transferred to one of the SMTP server processes created for this service. Specifying PORT=0 disables the current service.
STACKSIZE	Specifies the thread stack size of the server. The purpose of this option is to reduce the chances of the server running out of stack when processing deeply nested MIME messages (several hundreds of levels of nesting). Note that these messages are in all likelihood spam messages destined to break mail handlers. Having the server fail protects other mail handlers farther down the road.

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 4-27.

Table 4-27 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.
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.

SMS Channel Option File

The Messaging Server SMS (Short Message Service) channel is a one-way email to SMS gateway. Mail can be sent to an SMS gateway, but handling of SMS notifications (that is, replies and delivery receipts) and origination of email from SMS users (mobile to email) is presently not supported. The channel converts enqueued email messages to SMS messages. This conversion process includes handling of multipart MIME messages as well as character set translation issues.

The generated SMS messages are submitted to a Short Message Service Centre (SMSC) using the Short Message Peer to Peer (SMPP) protocol. Specifically, SMPP V3.4 is used over a TCP/IP connection to the SMSC's SMPP server. Operating in this capacity, the channel functions as a External Short Message Entity (ESME).

For more information about the SMS channel, see the *Sun Java System Messaging Server Administration Guide*.

An option file may be used to control various characteristics of the SMS channel. The channel options are stored in a text file in the *msg_svr_base*/config/ directory. 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:

msg_svr_base/config/sms_mway_option

Format of the File

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
```

A sample option file named `sms_option.sample` is distributed with Sun Java System Messaging Server. Copy this option file and use it as a starting point.

Available Options

The SMS channel contains a number of options which divide into four broad categories: email to SMS conversion, SMS fields, SMPP protocol, and localization. These categories and their corresponding options are detailed in the following sections.

Email to SMS Conversion

The email to SMS conversion options control the email to SMS conversion process. In general, a given email message may be converted into one or more SMS messages. These options are described in Table 4-28.

Table 4-28 SMS Channel Options: Email to SMS Conversion

Option	Description
`GATEWAY_NOTIFICATIONS`	Specify whether or not to convert email notification messages to SMS messages. Default: 0
`MAX_MESSAGE_PARTS` (Integer)	Maximum number of message parts to extract from an email message. When converting a multi-part email message to an SMS message, only the first `MAX_MESSAGE_PARTS` 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 generated the SMS message. Note that an email message containing both text and an attachment will typically consist of two parts. Note further that only message parts of type text are converted. All other MIME content types are discarded.

Table 4-28 SMS Channel Options: Email to SMS Conversion *(Continued)*

Option	Description
MAX_MESSAGE_SIZE (Integer, >=10)	Maximum number of bytes to extract from an email message. 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. For example, MAX_MESSAGE_SIZE=960 and MAX_PAGE_SIZE=160 implies MAX_PAGES_PER_MESSAGE=6. The two different options exist 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 the *Sun Java System Messaging Server Administration Guide*. 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)	Maximum number of bytes to allow into a single SMS message. By default, a value of 160 bytes is used.
MAX_PAGES_PER_MESSAGE (Integer, 1-255)	Maximum number of SMS messages to generate for a given email message. 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. By default, MAX_PAGES_PER_MESSAGE is set to the larger of 1 or MAX_MESSAGE_SIZE divided by MAX_PAGE_SIZE.
ROUTE_TO	Route SMS messages to the specified IP host name.

Table 4-28 SMS Channel Options: Email to SMS Conversion *(Continued)*

Option	Description
SMSC_DEFAULT_CHARSET (string)	Default character set used by the SMSC. The character set names in the *msg_svr_base*/imta/confic/charsets.txt file are used. US-ASCII is the default. When processing an email message, the header lines and text message parts are first decoded and then converted to Unicode. Next, the data is converted to either the SMCS's default character set or USC2, as follows: • 1—The SMSC default character set is used whenever possible. When the originating email message contains glyphs not in the SMSC default character set, then the UCS2 character set is used. • 0—The SMSC default character set is always used. Glyphs nor available in that character set are represented by mnemonics (for example, "AE" for AE-ligature).
USE_HEADER_FROM	Set this option to allow the From: address to be passed to the SMS channel. The value indicates where the From: address is taken from and what format it will have. 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 Default: 0
USE_HEADER_PRIORITY (0 or 1)	Controls the use of priority information from the email message's header (RFC822 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 RFC822 Priority: header line, specify USE_HEADER_PRIORITY=0. The default is USE_HEADER_PRIORITY =1. See the description of the DEFAULT_PRIORITY option for further information on the handling the SMS priority flag.

Table 4-28 SMS Channel Options: Email to SMS Conversion *(Continued)*

Option	Description
USE_HEADER_REPLY_TO (0 or 1)	Controls the use of Reply-to: header lines when generating SMS source addresses. When SET_SMS_SOURCE_ADDRESS=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. This is the reason why, by default, USE_HEADER_REPLY_TO=0.
USE_HEADER_RESENT (0 or 1)	Controls the use of Resent-*: header lines when generating originator information. When SET_SMS_SOURCE_ADDRESS=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; hence, why this option has the default value of 0.
USE_HEADER_SENSITIVITY (0 or 1)	Controls the use of privacy information from the email message's header (RFC822 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 RFC822 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)	Specifies that the UCS2 character set is to be used in SMS messages when applicable. The default behavior is to use the UCS2 character set, 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.

SMS Gateway Server Option

The SMS Gateway Server Option specifies the Gateway profile. shows. This option is described in Table 4-29.

Table 4-29 SMS Channel Options: SMS Gateway Server Option

Option	Description
GATEWAY_PROFILE	Match the gateway profile name configured in the SMS Gateway Server's configuration file, sms_gateway.cnf.

SMS Fields

The SMS fields options control SMS-specific fields in generated SMS messages. These options are described in Table 4-30.

Table 4-30 SMS Channel Options: SMS Fields

Option	Description
DEFAULT_DESTINATION_NPI (Integer, 0-255)	Default NPI for SMS destination addresses. By default, destination addresses are assigned an NPI (Numeric Plan Indicator) value of zero. With this option, an integer value in the range 0 to 255 may be assigned. Typical NPI values include: 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.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).

Table 4-30 SMS Channel Options: SMS Fields *(Continued)*

Option	Description
DEFAULT_DESTINATION_TON (Integer, 0-255)	Default TON for SMS destination addresses. By default, destination addresses are assigned a TON (Type of Number) designator value of zero. With this option, an alternate integer value in the range of 0 to 255 may be assigned. Typical TON values include: 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)	Default priority setting for SMS messages. All SMS messages have a mandatory priority field. The interpretation of SMS priority values is described in Table 4-31. 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 RFC822 Priority: header line, then the priority specified in that header line is instead used to set the priority of the resulting SMS message. Specifically, the results are as follows: 0—The SMS priority flag is always set in accord with the DEFAULT_PRIORITY option. The RFC822 Priority: header line is always ignored. 1 (default)—The originating email message's RFC822 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. In translating RFC822 Priority: header line values to SMS priority flags, the mappings used are described in Table 4-32.

Table 4-30 SMS Channel Options: SMS Fields *(Continued)*

Option	Description
DEFAULT_PRIVACY (Integer, -1, 0-255)	Default privacy value flag for SMS messages. Whether or not to set the privacy flag in an SMS message, and what value to use is controlled by the DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY options. By default, a value of -1 is used for DEFAULT_PRIVACY. The results from the combination of DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY values are described in Table 4-33. The SMS interpretation of privacy values is as follows: • 0—Unrestricted • 1—Restricted • 2—Confidential • 3—Secret • >=4—Undefined To translate Sensitivity: header line values to SMS privacy values, the following mapping is used: • Personal—1 (Restricted • Private—2 (Confidential)
DEFAULT_SERVICE_TYPE (String, 055 bytes)	SMS application service associated with submitted SMS messages. By default, no service type is specified (that is, a zero length string). Some common service types are: CMT (cellular messaging), SPT (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)	Default SMS source address to use for SMS messages generated from email messages. Note that the value specified with this option is typically overridden by the email message's originator address when SET_SMS_SOURCE_ADDRESS=1. The default is no source address is specified (0 length string).
DEFAULT_SOURCE_NPI (Integer, 0-255)	Default NPI for SMS source addresses. By default, source addresses are assigned an NPI value of zero. With this option, an alternate integer value in the range of 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_NPI option for a list of typical NPI values.

Table 4-30 SMS Channel Options: SMS Fields *(Continued)*

Option	Description
DEFAULT_SOURCE_TON (Integer, 0-255)	Default TON for SMS source addresses. By default, source addresses are assigned a TON designator value of zero. With this option, an alternate integer value in the range of 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_TON option for a list of typical TON values.
DEFAULT_VALIDITY_PERIOD (String, 0-252 bytes)	Default validity period for SMS messages. This option specifies a different relative validity period. By default, SMS messages are given no relative validity period, using the SMSC's default value. Values may be specified in units of seconds, minutes, hours, or days: *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.
DEFAULT_ADDRESS_NUMERIC (0 or 1)	Reduce the destination SMS address to only the characters 0-9. This option strips 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 performed before any destination address prefix is added via the DESTINATION_ADDRESS_PREFIX option.
DESTINATION_ADDRESS_PREFIX (String)	Text string with which to prefix destinationSMS addresses. 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.

Table 4-30 SMS Channel Options: SMS Fields *(Continued)*

Option	Description
PROFILE (String)	Specifies the SMS profile to use 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.
SET_SMS_SOURCE_ADDRESS (0 or 1)	Set the SMS source address to the originator address of the email message. Use of this option forces the SMS source address TON to be set to alphanumeric (0x05), and the SMS source address to be an originator address extracted from the email message. As email messages may have a number of originator addresses, the particular address chosen is the one most likely to be the address to which any replies should be directed. Consequently, the choice is made from one of the seven header lines described in Table 4-34, listed in order of decreasing preference: The selected address is reduced to just its local and domain parts; that is, any source route, phrase, or comments are stripped from the address. Furthermore, if the length of the reduced address exceeds 20 bytes, it will be truncated to 20 bytes. When none of the seven listed header lines are suitable, the default source SMS address is instead used as specified with the DEFAULT_SOURCE_ADDRESS option. In that case, the TON is set as per the DEFAULT_SOURCE_TON. To enable this option, specify SET_SMS_SOURCE_ADDRESS=1. By default, this option is enabled.
USE_SAR (0 or 1)	Sequence multiple SMS messages using the SMS sar_fields. 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.

Table 4-31 describes the interpretation of the priority field for the DEFAULT_PRIORITY option.

Table 4-31 Priority Fields for DEFAULT_PRIORITY

Value	GSM	TDMA	CDMA
0	Non-priority	Bulk	Normal
1	Priority	Normal	Interactive
2	Priority	Urgent	Urgent
3	Priority	Urgent	Emergency

Table 4-32 describes the mappings used in translating Priority: header line values to SMS priority flags for the DEFAULT_PRIORITY option.

Table 4-32 Mappings for Priority Flags

RFC822	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)

The results from the combination of DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY values are described in Table 4-33.

Table 4-33 Results from DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY Values

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. RFC822 Sensitivity: header lines are always ignored.
-1 (default	1 (default)	The SMS message's privacy flag is only set when the originating email message has an RFC822 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.

Table 4-33 Results from DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY Values

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
n >= 0	1	The SMS message's privacy flag is set to correspond to the originating email message's RFC822 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.

Table 4-34 describes the seven header lines used with the SET_SMS_SOURCE_ADDRESS option, their restrictions and SMS source address TON (if applicable) in decreasing preference.

Table 4-34 SET_SMS_SOURCE_ADDRESS Header Restrictions

Email message field	Restrictions	TON
1. Resent-reply-to:	Requires USE_HEADER_RESENT=1 and USE_HEADER_REPLY_TO=1	
2. Resent-from:	Requires USE_HEADER_RESENT=1	
3. Reply-to:	Requires USE_HEADER_REPLY_TO=1	0x05
4. From:		
5. Resent-sender:	Requires USE_HEADER_RESENT=1	
6. Sender:		
7.Envelope From:		
8. DEFAULT_SOURCE_ADDRESS	Used as a last resort (that is, when the envelope From: address is empty)	As per DEFAULT_SOURCE_TON

SMPP Protocol

The SMPP protocol options are associated with the use of the SMPP protocol over TCP/IP. 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. These options are described in Table 4-35.

Table 4-35 SMS Channel Options: SMPP Protocol

Option	Description
ESME_ADDRESS_NPI (Integer, 0-255)	ESME NPI to specify when binding to the SMPP server. By default, bind operations 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)	ESME TON to specify when binding to the SMPP server. By default, bind operations 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)	IP address of the host running Messaging Server. 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, 0-9 bytes)	Password to present when binding to the SMPP server. If a password is required, then specify it with this option. By default, a zero-length password string is presented.
ESME_SYSTEM_ID (String, 0-15 bytes)	System identification to present to the SMSC when binding. If a password is required, then specify it with this option. By default, a zero-length password string is presented.
ESME_SYSTEM_TYPE (String, 0-12 bytes)	System type for the MTA to present to the SMSC when binding. By default, no system type is specified (that is, a zero-length string is used).

Table 4-35 SMS Channel Options: SMPP Protocol *(Continued)*

Option	Description
MAX_PAGES_PER_BIND (Integer, >=0)	Maximum number of SMS messages to submit during a single session with an SMPP server. 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 unbinds, closes the TCP/IP connection, re-connects, and then rebinds. By default, a value of 1024 is used for MAX_PAGES_PER_BIND. Note that the channel also detects ESME_RTHROTTLED errors and adjusts MAX_PAGES_PER_BIND during a single run of the channel accordingly.
REVERSE_ORDER (0 or 1)	Transmission sequence of multi-part SMS messages. 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)	Maximum number of simultaneous SMPP server 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)	TCP port on which the SMPP server listens. The TCP port 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.
SMPP_SERVER (String, 1-252 bytes)	Host name of the SMPP server to which to connect. By default, the IP host name of the SMPP server to which to connect 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 overrides that specified in the channel definition. When specifying an IP address, use dotted decimal notation; for example, 127.0.0.1

Table 4-35 SMS Channel Options: SMPP Protocol *(Continued)*

Option	Description
TIMEOUT (Integer, >=2)	Timeout for completion of read and write actions with the SMPP server. 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 2 seconds.

Localization

The Localization options allow for localization of text fields inserted into SMS messages. These options are described in Table 4-36. In constructing SMS messages, the SMS channel has a number of fixed text strings it places into those messages. These strings, for example, introduce the email's From: address and Subject: header line. With the channel options described below, versions of these strings may be specified for different languages and a default language for the channel then specified. This section of the option file appears as follows:

```
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.

Table 4-36 SMS Channel Options: Localization

Option	Description
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	Formatting string for delivery delay notifications
DSN_FAILED_FORMAT	Formatting string for delivery failure notifications
DSN_RELAYED_FORMAT	Formatting string for relay notifications.
DSN_SUCCESS_FORMAT	Formatting string to successful delivery notifications.
FROM_FORMAT (String, 0-252 bytes)	Text to display indicating the originator of the email message. The default global value is the US-ASCII string "$a" which substitutes in the originator's email address.
FROM_NONE (String, 0-252 bytes)	Text to display when there is no originator address to display. The default global value is an empty string. Note that normally, this option is never used as sites typically reject email messages which lack any originator address.
LANGUAGE (String, 0-40 bytes)	Language group from which to select text fields. If not specified, the language is derived from the host's default locale specification. If the host's locale specification is not available or corresponds to "C" then i-default is used. (i-default corresponds to "English text intended for an international audience.")
LINE_STOP (String, 0-252 bytes)	Text to place at the end of each line extracted from the email message. The default global value is the US-ASCII space character.
NO_MESSAGE (String, 0-252 bytes)	Text to indicate that the message contains no content. The default global value is the US-ASCII string "[no message]".
REPLY_PREFIX (String, 0-252 bytes)	Reserved for future use. The default global value is the US-ASCII string "Re: ".
SUBJECT_FORMAT (String, 0-252 bytes)	Formatting template to format the content of the Subject: header line for display in the SMS message. The global default value for this option is the US-ASCII string "($s)". See 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 to display when no subject exists for the email message, or the Subject: header line's value is an empty string. The default global value for this option is the empty string.

Miscellaneous

Debug: Enable verbose debug output.

Messaging Multiplexor Configuration

This chapter describes the Messaging Multiplexor configuration. This chapter contains the following sections:

NOTE	To configure HTTP user mailboxes (for example, Messenger Express), see the chapter "Configuring and Administering Multiplexor Support" in the *Sun Java System Messaging Server Administration Guide*.

Encryption (SSL) Option

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

When SSL is enabled, the MMP IMAP supports both STARTTLS on the standard IMAP port and IMAP+SSL on port 993. The MMP can also be configured to listen on port 995 for POP+SSL.

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

To enable SSL encryption for SMTP proxy services, edit the `SmtpProxyAService.cfg` file.

By default, SSL is not enabled since the SSL configuration parameters (Table 5-1) are commented out. Install a certificate as documented in the *Sun Java System Messaging Server Administration Guide*. To enable SSL, un-comment and set the following parameters:

Table 5-1 SSL Configuration Parameters

Parameter	Description
`SSLBacksidePort`	Port number to which the MMP will try to connect on the store servers for SSL. If this parameter is not set, the MMP will not use SSL when connecting to the store. There are no default values, but ports 993 and 995 are recommended for IMAP and POP, respectively. This parameter does not apply to SMTP proxy.
`SSLCacheDir`	SSL session cache directory. The recommended value is the *msg_svr_base*`/config` directory.
`SSLCertFile`	This has been replaced by the option `SSLCertPrefix`.
`SSLCertNicknames`	Nicknames of the certificates in the SSL certificate database to offer as the server certificate. The recommended value is `Server-Cert`.
`SSLCertPrefix`	Filename prefix to the SSL certificate database file. The certificate database file must be in the directory specified by the `SSLCacheDir` setting. The recommended value is ““.

`SSLEnable`	Whether or not to enable SSL. If set to “True”, “Yes” or “1”, Multiplexor will activate the STARTTLS (for IMAP, SMTP) or STLS (for POP) command. To activate SSL on separate ports, this must be set in addition to the SSLPorts option.
	If SSL is enabled, all of the following variables must be set. You can specify an empty parameter with empty quotes (““).
	`SSLPorts`
	`SSLCertPrefix`
	`SSLKeyPrefix`
	`SSLKeyPasswdFile`
	`SSLCertNicknames`
	The default is no (SSL is not enabled).

`SSLKeyPrefix`	Key database file location (defined when you obtained a certificate for this server). Multiplexor requires a private key corresponding to its SSL server certificate. The location specified here should be absolute, not relative to the Multiplexor installation directory. The recommended value is *msg_svr_base*`/config/key3.db.` Be sure to protect this file so only the multiplexor and other authorized servers can read it.

Table 5-1 SSL Configuration Parameters *(Continued)*

Parameter	Description
SSLKeyPasswdFile	File location for the passwords that protect access to the private key file. Passwords may be null if the key is not password-protected. The default is *msg_svr_base*/config/sslpassword.conf.
SSLPorts	Ports on which SSL will be turned on (accepted SSL connections). Syntax is: [IP ":"] PORT ["|" [IP ":"] PORT] For example: 993|127.0.0.1:1993 means connections to any IP on port 993 and localhost on port 1993 get SSL on accept. There are no default values, but ports 993 and 995 are recommended for POP and IMAP, respectively. Note that even if you set a port, the MMP will not actually accept connections to that port until it is included in the ServiceList (see "Multiplexor Configuration Parameters" on page 318). If this parameter is not set, and SSLEnable is set to "true" or "yes," then only IMAP STARTTLS is enabled.
SSLSecmodFile	Security module database file location. If you have hardware accelerators for SSL ciphers, this file describes them to the Multiplexor. The recommended value is secmod.db.

Multiplexor Configuration

This section describes how to configure the Messaging Multiplexor.

Multiplexor Configuration Files

To configure the Multiplexor, you must manually edit the configuration parameters in the Multiplexor configuration files, which are listed below in Table 5-2.

Table 5-2 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.

Table 5-2 Messaging Multiplexor Configuration Files *(Continued)*

File	Description
`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 the *Sun Java System Messaging Server Administration Guide*.
`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.

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

The following configuration parameters are defined in the `AService.cfg` file:

- `ServiceList`
- `LogDir and LogLevel`
- `NumThreads`
- `BeTheUser and BeTheGroup`

For descriptions of these parameters, see ""Multiplexor Configuration Parameters" on page 301.

The Multiplexor configuration files are stored in the *msg_svr_base*/mmp-*hostname* directory, where *msg_svr_base* is the directory where you installed the Messaging Server and mmp-*hostname* is the subdirectory named after the MMP instance.

Multiplexor Configuration Parameters

You control how the MMP operates by specifying various configuration parameters in the MMP configuration files.

Table 5-3 describes the parameters you can set:

NOTE To allow configuration parameters for different instances to be specified in the same configuration file, all the parameters are preceded with "default:" to indicate the default section. See the ServiceList parameter in Table 5-3 for more information.

Table 5-3 Multiplexor Configuration Parameters

Variable	Description
AuthCacheSize AuthCacheTTL	The MMP can cache results of pre-authentication. The AuthCacheSize parameter defines the number of cache entries; AuthCacheTTL defines the length of time that entries are preserved in seconds. Lower values will reduce performance, but result in faster recognition or server password changes. Higher values will increase performance, but result in delayed recognition of server password changes. A higher setting for AuthCacheSize improves performance while using more memory. A lower setting reduces performance and reduces the amount of memory used. AuchCacheTTL controls how long a cachec entry remains in cache. Changes made to an entry in LDAP are not seen by the MMP until the entry's TTL has expired. If you wish to have password changes seen at least every 15 minutes by the MMP, then set this value to 900. These variables are only applicable when PreAuth is set to yes. The default AuthCacheSize is 10,000; the default AuthCacheTTL is 900. This options does not apply to SMTP proxy.
AuthenticationLdapAttributes	A space-separated list of additional LDAP attributes to look up and pass to the third-party authentication server specified by the AuthenticationServer option.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
AuthenticationServer	This specifies the hostname and port for a third-party authentication service to use with the MMP. The recommended value is 127.0.0.1:56 when a third-party authentication service is available on the same machine as the MMP. For developer instructions and SDK see the directory *msg_svr_base*/examples/tpauth. When not set, the MMP will authenticate via LDAP. This parameter is ignored unless the PreAuth option is set to yes. This parameter does not apply to the SMTP proxy.
AuthService AuthServiceTTL	If AuthService is set to yes and AuthServiceTTL is non-zero, the MMP will allow queries about who is currently logged into the MMP, for the purpose of POP before SMTP relay authentication. AuthServiceTTL represents the amount of time in seconds that an authentication record is kept valid. The default for AuthService is no; the default AuthServiceTTL is -1. The AuthService parameter should almost never be turned on globally; you should configure this by virtual domain. Setting the AuthService parameter to yes permits probing of the AuthService cache with the xqueryauth *ip-address* command over the POP protocol. For POP before SMTP service, AuthServiceTTL should be set to a value greater than 0 in the PopProxyAService.cfg file. For all other MMP proxies (SMTP and IMAP), AuthServiceTTL should be omitted or set to -1. By default, the AuthServiceTTL parameter is found only in the PopProxyAService.cfg configuration file.
BacksidePort	Port on which to connect to message store server. This parameter lets you run a multiplexor and a store server on the same machine, with the store server on a different port. You might want to do this if you want a flat configuration—that is, if you want to run Multiplexors on all machines. This option does not apply to SMTP proxy. The SmtpRelays parameter provides equivalent functionality for the SMTP proxy. The default is 110 for POP3; 143 for IMAP (the standard ports).
Banner	Banner replacement string. The MMP will use the string you specify for its greeting line. The default banner string contains the software name and version information.
BeTheUser and BeTheGroup	BeTheUser and BeTheGroup are the user ID and group ID of the MMP, respectively, once it has started listening for connections. These values are set by the Messaging Server configure installation program. These variables are applicable to UNIX only and are ignored on Windows platforms. The BeTheUser and BeTheGroup parameters are only found in the AService.cfg configuration file.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
BGMax BGPenalty BGMaxBadness BGDecay BGLinear BGExcluded	BadGuys configuration parameters. When an authentication failure occurs from a particular client IP address, subsequent authentication attempts from that IP address are treated as "BadGuys" and are delayed. If an authentication failure is followed by a successful authentication, the successful authentication is delayed, but the IP address ceases to be treated as a "BadGuy" for subsequent attempts. BGMax is the maximum number of BadGuys to keep track of simultaneously (default is 10,000). BGPenalty is the length of time in seconds added to a BadGuy's sentence if he/she fails authentication (default is 2). BGMaxBadness is the maximum penalty in seconds for authentication failure (default is 60). BGDecay represents the time in seconds it takes for a BadGuy's penalty to be forgiven (default is 900). BGLinear defines whether a BadGuy's penalty decays linearly over time, or is a step function on expiration (default is no, which means the penalty decays as a step function on expiration). BGExcluded represents a list of excluded IP/mask pairs, or the name of a file to read for these pairs. These client addresses will not be penalized for authentication failure (there is no default value). The BadGuys parameters are only applicable when PreAuth is set to yes. These parameters do not apply to SMTP proxy.
BindDN BindPass	Distinguished Name and password used to authenticate to the Directory Server. The BindDN must have privileges to access the BaseDN as specified by the LdapURL. The Messaging Server default directory ACIs require a bind to authenticate users against the Directory Server. These options can be found in the ImapProxyAservice.cfg and PopProxyAservice.cfg configuration files. These parameters do not apply to SMTP proxy.
CanonicalVirtualDomainDelim	Canonical virtual domain delimiter. The character used by the MMP to separate the user ID from the appended virtual domain when talking to the message store server and formatting queries for the LDAP server. The default is @, so user IDs passed to LDAP and the message store servers have the form userid@virtual.domain. This parameter does not apply to SMTP proxy.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`Capability`	Capability replacement string. The MMP will use the string you specify for `Capability` instead of its default (own) capability to tell IMAP clients what it (or the servers behind it) can do. This variable has no effect in POP3. There is no need to adjust this string if the backend IMAP servers are entirely Sun Java System servers from the same version of the messaging server installer. Otherwise, it is important to specify a capability list that includes only the features supported by all the backend IMAP servers. The appropriate string can be determined by telnetting to port 143 on each kind of backend server and entering the command `c capability`. This lists only the capabilities present on all backend IMAP servers. The default Capability string is as follows (with no line breaks): `"IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN BINARY LANGUAGE XSENDER X-NETSCAPE XSERVERINFO"` When Messaging Server 5.2 backend mail stores are used, the `BINARY` option should be omitted. This parameter does not apply to SMTP proxy. The `EhloKeywords` parameter provides a semi-equivalent function for the SMTP proxy.
`CertmapDN`	This option is equivalent to `UserGroupDN` and is deprecated in favor of the new name (`UserGroupDN` takes precedence over this setting).
`CertMapFile`	The name of the certmap file (for SSL client-cert-based authentications). There is no default. The recommended setting is *msg_svr_base*`/config/certmap.conf`
`ClientLookup`	Performs a DNS reverse lookup on the client IP address when set to `yes`. The reverse lookup is performed unconditionally, so the SMTP relay server does not need to perform it. This option may be set on a per hosted domain basis. The `ClientLookup` parameter provides a performance benefit for SMTP, but has no benefit when used with POP or IMAP. Note that a DNS lookup is performed regardless of this setting if hostnames are used in a global TCPAccess filter or a per-domain or per-user access filter. This option defaults to no. For example: `default:ClientLookup yes`

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
ConnLimits	Limits the number of simultaneous connections permitted from a single client IP address. A comma-separated list of entries in the following form: IP “|” MASK “:” NUM or the path and name of a specific file containing one or more of these entries; each entry on its own line. The entries should be listed from the most specific IP-MASK pairs to the least specific. The default is 0.0.0.0|0.0.0.0:20
CRAMs	Boolean indicating whether or not to enable Challenge-Response Authentication Mechanisms (CRAMs) including APOP and CRAM-MD5. For this to work, passwords must be stored in LDAP in plain text format and the BindDN must have read access to the userPassword attribute. The default is no. This parameter does not apply to SMTP proxy.
DefaultDomain	When POP and IMAP users authenticate, they typically provide an unqualified user ID (a user ID without a domain portion). The value of the DefaultDomain parameter is appended to unqualified user IDs. When used as an MMP virtual domain parameter, this allows a single MMP server with multiple IP addresses to support unqualified user IDs for multiple hosted domains. This may also be set as a service-wide parameter. This parameter does not apply to SMTP proxy.
EhloKeywords	A list of EHLO extension keywords for the proxy to pass through to the client, in addition to the default set. The MMP removes 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 supports the following keywords (there is no need to list them in this option): 8BITMIME, PIPELINING, 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: default: EhloKeywords TURN This parameter is found only in the SmtpProxyAService.cfg file.
FailoverTimeout	If a connection to an SMTP relay fails, the MMP avoids trying that relay for a number of minutes equivalent to the failover time-out. For example, if the failover time-out is 10 seconds, and a relay fails, the MMP does not try that relay again for 10 minutes. The default is 10 seconds.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
HostedDomains	Boolean, whether to support HostedDomains. If you are using the Sun Java System Messaging Server directory schema (Sun ONE LDAP Schema, v1 or Sun ONE LDAP Schema, v2), this should be set to the default "Yes." If set to no, then the MMP assumes the server supports only one domain and LdapUrl points to a directory subtree containing all users supported by the server, each user with a unique UID. Setting HostedDomains to "no" is not recommended as even a small company is likely to eventually go through a name change or acquisition where support for multiple domains would be helpful. When set to yes, the MMP honors the following MTA options in the *msg_svr_base*/config/option.dat file: LDAP_SCHEMALEVEL LDAP_DOMAIN_FILTER_SCHEMA1 LDAP_DOMAIN_FILTER_SCHEMA2 LDAP_ATTR_DOMAIN1_SCHEMA2 LDAP_ATTR_DOMAIN2_SCHEMA2 LDAP_GLOBAL_CONFIG_TEMPLATES LDAP_ATTR_DOMAIN_SEARCH_FILTER LDAP_DOMAIN_ATTR_BASEDN LDAP_DOMAIN_ATTR_CANONICAL LDAP_DOMAIN_ATTR_ALIAS These settings may be used to enable Sun ONE LDAP Schema, v2 with the MMP. Defaults to Yes. This parameter does not apply to SMTP proxy.
LdapCacheSize LdapCacheTTL	The MMP can cache results of user searches. The LdapCacheSize parameter defines the number of cache entries; LdapCacheTTL defines the length of time the entries are preserved in seconds. Lower values will reduce performance, but result in faster recognition of LDAP user configuration changes. Higher values will increase performance, but result in delayed recognition of LDAP user configuration changes. The default LdapCacheSize is 10,000; the default LdapCacheTTL is 900. These parameters do not apply to SMTP proxy.
LdapRefreshInterval	Seconds that the MMP will keep a connection open to the LDAP server. When the MMP notices the refresh interval has passed, the MMP will close the LDAP connection and open a new one. The default is 2100 (35 minutes).

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`LdapUrl`	Pointer to the top of the site's DC directory tree, if `HostedDomains` is set to `yes` (default). If `HostedDomains` is set to no, then `LdapUrl` points to a directory subtree containing all users supported by the server. This parameter must be set in order for the MMP to operate correctly. SSL (LDAPS) is supported, but the SSL configuration must also be correct, and SSL-enabled. To enable failover, the host part of the URL may be a space-separated list of hosts. Be sure to enclose the entire URL in double-quotes if it contains a space. For example: `"ldap://ldap1 ldap2/o=internet"` The default is `ldap://`*localhost*`/o=internet`. This parameter does not apply to SMTP proxy.
`LogDir` `LogLevel`	`LogDir` is the directory in which the MMP creates log files. If you specify a directory that does not exist, no log file is created. Log file names are distinguished by their specific service; for example, an IMAP log file would have the format `ImapProxy_`*yyyymmdd*`.log` . `LogLevel` represents the logging verbosity level—the amount of information written into log files. You can specify a number from 0 through 10, with 10 representing the highest level of verbosity. The higher the level, the more information in the log. `LogDir` and `LogLevel` are present in all configuration files: `ImapProxyAService.cfg`, `PopProxyAService.cfg`, `AService.cfg`, and `SmtpProxyAService.cfg`. The default `LogDir` is *msg_svr_base*`/data/log` and the default `LogLevel` is 1.
`MailHostAttrs`	Space-separated list of LDAP attributes identifying the user's mail host. Multiplexor tries each attribute returned by the search in the order specified by the list. The default is `mailHost`. This parameter does not apply to SMTP proxy.
`NumThreads`	The maximum number of worker threads to allocate. If the machine has multiple CPUs, running the Multiplexor with worker threads will improve performance. The optimal number of work threads is the number of processors on the machine. For example if your machine has two CPUs, specify 2. This parameter is only found in the `AService.cfg` configuration file. The default is 1.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
PopBeforeSmtpKludgeChannel	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. For example: default:PopBeforeSmtpKludgeChannel tcp_intranet This parameter is only found in the SmtpProxyAService.cfg configuration file.
PreAuth	Enables pre-authentication by the MMP. When PreAuth is set to yes, a user is authenticated against the LDAP server before a connection is made to the backend mailstore server. When PreAuth is set to no, the MMP connects to the backend mailstore server and simply replays the authentication information. Because of the additional authentication step, PreAuth reduces the overall performance, but protects the backend mailstore servers from denial-of-service attacks by unapproved users. PreAuth is mandatory for the POP-before-SMTP and BadGuys features of the MMP. When using HostedDomains, the mailAccessProxyPreAuth attribute in the domain node in the LDAP server overrides this option. The default is no. This parameter does not apply to SMTP proxy.
ReplayFormat	Printf-style format string that says how to construct the user ID for replay to the Message Store server. Valid escape sequences are: %U (userid only) %V (virtual domain only) %A[attr] (value of user's attribute "attr") For example, %A[uid]@%V for a user with joe as the user ID and domain=siroe.com would yield: joe@siroe.com. When using HostedDomains, the mailAccessProxyReplay attribute in the domain node in the LDAP server overrides this option. The default is %U. This parameter does not apply to SMTP proxy.
RestrictPlainPasswords	When set to yes, this will forbid use of plaintext passwords unless an SSL/TLS security layer is active. Defaults to no.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`SearchFormat`	A printf-style format string with which to construct Users/Groups LDAP queries for the user's mailhost when virtual domains are enabled. valid escape sequences are: %s (userid+virtualdomain) %U (userid only) %V (virtual domain only) %C (client IP address) %S (server IP address) %D (client cert DN) The default value is `uid=%U` if `HostedDomains` is yes, and `uid=%s` if `HostedDomains` is no. Note that when using `HostedDomains`, the `inetDomainSearchFilter` attribute in the domain node in the LDAP server overrides this option. This parameter does not apply to SMTP proxy.
`ServerDownAlert`	IMAP only. String returned to client in an IMAP ALERT message when the MMP cannot connect to a user's store server. The default string is "Your IMAP server appears to be temporarily out of service."
`ServiceList`	Specifies which services to start and the ports/interfaces on which the MMP will listen for those services. Services are listed all on a single line in the following format: *DLLNAME* `[ "|" ` *INSTANCENAME* `[ "|" ` *SECTION* `]] "@" ` *HOSTPORT* `[ "|" ` *HOSTPORT* `]` Where *DLLNAME* is the absolute pathname and filename to the AService DLL you want to load (minus the DLL file extension, `.so`, `.dll`, etc.). If no *DLLNAME* is specified or the one(s) specified cannot be loaded and initialized, the AService daemon will exit. Customer-supplied DLLs (shared libraries) are not supported. The *INSTANCENAME* represents the name of the configuration file to use for IMAP, POP, or SMTP services (minus the `.cfg` extension, so the defaults are `ImapProxyAService`, `PopProxyAService`, and `SmtpProxyAService`, respectively). *INSTANCENAME* can also take an optional *SECTION* parameter which allows configuration for different instances to be stored in the same config file. Use of *SECTION* is not recommended and it will be removed in a future release. The default *SECTION* is default. The `ServiceList` parameter is only found in the `AService.cfg` configuration file. The default `ServiceList` entry is shown below (all on one line): *msg_svr_base*`/lib/ImapProxyAService@143|993` *msg_svr_base*`/lib/PopProxyAService@110`

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
SmtpProxyPassword	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 from the SMTP channel option file. For example: default:SmtpProxyPassword *password* This parameter is only found in the SmtpProxyAService.cfg configuration file.
SmtpRelays	A space-separated list of SMTP relay server hostnames (with optional port) to use for round-robin relay. These relays must support the XPEHLO extension. This option is mandatory with no default. For example: `default:SmtpRelays sesta:485 gonzo mothra` This parameter is only found in the SmtpProxyAService.cfg configuration file.
SpoofMessageFile	The file to use for POP3 inbox spoofing. The MMP can imitate a base-functionality POP3 server in case it can't connect to a client's store machine. In such a situation, the MMP creates an inbox for the user and places this one message into it. The format of the message contained in this file should conform to RFC 822 (including the final '.'). By default, there is no spoof message file.
StoreAdmin StoreAdminPass	StoreAdmin represents the user name of the store administrator for proxy authentication necessary to support SSL client certificates and RFC 2595-style proxy authentication. There is no default for StoreAdmin or StoreAdminPass. This parameter does not apply to SMTP proxy.
TCPAccess	Wrap-style filters that describes TCP access control for the MMP (globally). See "Configuring Client Access to POP, IMAP, and HTTP Services" in the "Configuring Security and Access Control" chapter of the *Sun Java System Messaging Server Administration Guide* for the syntax description of this option. Defaults to NULL.
TCPAccessAttr	Per-user attribute that contains a wrap-style filter describing the TCP access control for the user. Defaults to mailAllowedServiceAccess.
Timeout	Session timeout in seconds. To be standards-compliant, the value of this parameter must not be set lower than 1800 seconds (30 minutes) for IMAP, 600 seconds (10 minutes) for POP or SMTP. The default is 1800 seconds.

Table 5-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`UserGroupDN`	This specifies the baseDN for user, group and domain searches in Sun ONE LDAP Schema, v2 mode. It is also used for client certificate mapping lookups in Sun ONE LDAP Schema, v1 mode.
`VirtualDomainDelim`	String of acceptable virtual domain delimiters. Any character in this string will be treated as a domain delimiter in a user ID received by the MMP. (The MMP searches user IDs from the end.) The default delimiter is `@`. This parameter does not apply to SMTP proxy.
`VirtualDomainFile`	The name of the file containing your virtual domain mapping. The recommended setting is *msg_svr_base*`/config/vdmap.cfg`. Uncomment this line in the configuration file to enable support for virtual domains.

Starting the Multiplexor

To start, stop, or refresh an instance of the Messaging Multiplexor, use the one of the following commands in Table 5-4 located in the *msg_svr_base*`/sbin` directory:

Table 5-4 MMP Commands

Option	Description
`start-mmp`	Starts the MMP (even if one is already running).
`stop-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.

Appendix A

Supported Standards

This appendix lists national, international, and industry standards related to electronic messaging and for which support is claimed by Sun Java System Messaging Server. Most of these are Internet standards, published by the Internet Engineering Task Force (IETF) and approved by the Internet Activities Board (IAB). Standards for documents from other sources are noted.

Several of the documents are listed with an obsolete status. These are included because they describe protocol features that were obsolete or replaced by later documents, but are still in widespread use.

Messaging

The following documents are relevant to national and international standards for messaging, specifically messaging structure.

Basic Message Structure

The structure of basic messages is explained in the documents listed in Table A-1.

Table A-1 Basic Message Structure

Standard	Status	Description
RFC 822 STD 11	Standard	David H. Crocker, University of Delaware, *Standard for the Format of ARPA Internet Text Messages*, August 1982.
RFC 1123	Standard	Robert Braden (Editor), *Requirements for Internet Hosts - Application and Support*, Internet Engineering Task Force, October 1989.
RFC 2822	Proposed Standard	P. Resnick (Editor), *Internet Message Format*, April 2001.

Access Protocols and Message Store

The documents listed in Table A-2 contain information about access protocols and message stores.

Table A-2 Access Protocols and Message Store

Standard	Status	Description
RFC 1730	Proposed Standard	Mark R. Crispin, (University of Washington), *Internet Message Access Protocol - Version 4*, December 1994.
RFC 1731	Proposed Standard	John G. Myers, (Carnegie-Mellon University), *IMAP4 Authentication Mechanisms*, December 1994.
RFC 1939	STD 53	John G. Myers (Carnegie-Mellon University) and Marshall T. Rose (Dover Beach Consulting), *Standard Post Office Protocol - Version 3*, May 1996.
RFC 1957	Information	R. Nelson, *Some Observations on Implementations of the Post Office Protocol (POP3)*, June 1996
RFC 2060	Proposed Standard	Mark Crispin (University of Washington), *Internet Message Access Protocol - Version 4rev1*, December 1996.
RFC 2061	Information	Mark R. Crispin (University of Washington), *IMAP4 Compatibility With IMAP2bis*, December 1996.
RFC 2062	Proposed Standard	Mark R. Crispin (University of Washington), *Internet Message Access Protocol - Obsolete Syntax*, December 1996.
RFC 2086	Proposed Standard	John G. Myers, *IMAP4 ACL Extension*, January 1997.
RFC 2087	Proposed Standard	John G. Myers, *IMAP4 QUOTA Extension*, January 1997.
RFC 2088	Proposed Standard	John G. Myers, *IMAP4 Non-Synchronizing Literals*, January 1997.
RFC 2180	Information	M. Gahrns, *IMAP4 Multi-Accessed Mailbox Practice*, July 1997.
RFC 2342	Proposed Standard	M. Gahrns, *IMAP4 Namespaces*, July 1997.
RFC 2359	Proposed Standard	John G. Myers, *IMAP4 UIDPLUS Extension*, June 1998.
RFC 2683	Information	B. Leiba, *IMAP4 Implementation Recommendations*, September 1999.
RFC 3206	Proposed Standard	R. Gellens, *SYS and AUTH POP Response Codes*, February 2002.
RFC 3501	Proposed Standard	M. Crispin, *Internet Message Access Protocol - Version 4rev1,* March 2003.
RFC 3516	Proposed Standard	L. Nerenberg, *IMAP4 Binary Content Extension*, April 2003

SMTP and Extended SMTP

The documents listed in Table A-3 contain information about Simple Mail Transfer Protocol (SMTP) and Extended SMTP.

Table A-3 SMTP and Extended SMTP

Standard	Status	Description
RFC 821 STD 10	Standard	Jonathan B. Postel, USC/Information Sciences Institute, *Simple Mail Transfer Protocol*, August 1982.
RFC 974 STD 14	Standard	C. Partridge, *Mail Routing and the Domain System*, January 1986.
RFC 1123 STD 3	Standard	R.T. Braden, *Requirements for Internet Hosts - Application and Support*, October 1989.
RFC 1428	Information	Greg Vaudreuil, Corporation for National Research Initiatives, *Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME*, February 1993.
RFC 1652	Draft Standard	John Klensin (United Nations University), Einar Stefferud (Network Management Associates, Inc.), Ned Freed (Innosoft), Marshall Rose (Dover Beach Consulting), David Crocker (Brandenburg Consulting), *SMTP Service Extension for 8bit-MIME transport*, July 1994.
RFC 1869 STD 10	Standard	John Klensin (United Nations University), Ned Freed (Innosoft), Marshall Rose (Dover Beach Consulting), Einar Stefferud (Network Management Associates, Inc.), David Crocker (The Branch Office), *SMTP Service Extensions*, November 1995.
RFC 1870 STD 10	Standard	John Klensin (United Nations University), Ned Freed (Innosoft), Keith Moore (University of Tennessee), *SMTP Service Extension for Message Size Declaration*, November 1995.
RFC 1985	Proposed Standard	J. De Winter, *SMTP Service Extension for Remote Message Queue Starting*, August 1996.
RFC 2034	Proposed Standard	Ned Freed, *SMTP Service Extension for Returning Enhanced Error Codes*, October 1996.
RFC 2442	Information	J. Belissent, *The Batch SMTP Media Type*, November 1998.
RFC 2476	Proposed Standard	R. Gellens, *Message Submission*, December 1998.
RFC 2821	Proposed Standard	J. Klensin (Editor), *Simple Mail Transfer Protocol*, April 2001.
RFC 2920 STD 60	Standard	Ned Freed, *SMTP Service Extension for Command Pipelining*, September 2000.
RFC 3028	Proposed Standard	T. Showalter, *Sieve: A Mail Filtering Language*, January 2001.
RFC 3207	Proposed Standard	P. Hoffman, *SMTP Service Extension for Secure SMTP over Transport Layer Security,* February 2002

Table A-3 SMTP and Extended SMTP *(Continued)*

Standard	Status	Description
RFC 3431	Proposed Standard	W. Segmuller, *Sieve Extension: Relational Tests*, December 2002
RFC 3461	Standard	*K. Moore, Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)*, January 2003. (Obsoletes RFC1891)
RFC 3462	Standard	G. Vaudreuil, *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages*, January 2003. Draft. (Obsoletes RFC1892)
RFC 3463	Standard	G. Vaudreuil, *Enhanced Mail System Status Codes*, January 2003. (Obsoletes RFC1893)
RFC 3598	Proposed Standard	K. Murchison, *Sieve Email Filtering -- Subaddress Extension*, September 2003

Message Content and Structure

The following documents specify message contents handling, most of which is covered by the Multipurpose Internet Mail Extensions (MIME). There are also several non-standard message content RFCs that are supported by the SIMS product, which are listed separately in Table A-4.

Table A-4 Message Content and Structure

Standard	Status	Description
RFC 1847	Proposed Standard	J. Galvin, S. Murphy, S. Crocker, N. Freed, *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*, October 1995.
RFC 2017	Proposed Standard	Ned Freed (Innosoft), Keith Moore (University of Tennessee), *Definition of the URL MIME External-Body Access-Type*, October 1996.
RFC 2045	Draft Standard	Nathaniel Borenstein (First Virtual Holdings) and Ned Freed (Innosoft), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, November 1996.
RFC 2046	Draft Standard	Nathaniel Borenstein (First Virtual Holdings) and Ned Freed (Innosoft), *MIME Part Two: Media Types*, November 1996.
RFC 2047	Draft Standard	Keith Moore (University of Tennessee), *MIME Part Three: Message Header Extensions for Non-ASCII Text*, November 1996.
RFC 2048	Policy	Ned Freed (Innosoft), John Klensin (MCI), Jon Postel (USC/Information Sciences Institute), *MIME Part Four: Registration Procedures*, November 1996.
RFC 2049	Draft Standard	Nathaniel Borenstein (First Virtual Holdings) and Ned Freed (Innosoft), *MIME Part Five: Conformance Criteria and Examples*, November 1996.
RFC 2231	Proposed Standard	N. Freed, K. Moore, *MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations*, November 1997.

Table A-4 Message Content and Structure *(Continued)*

Standard	Status	Description
RFC 2298	Proposed Standard	R. Fajman, *An Extensible Message Format for Message Disposition Notifications*, March 1998.

Delivery Status Notifications

The list of documents in Table A-5 describe delivery status notification.

Table A-5 Delivery Status Notifications

Standard	Status	Description
RFC 1891	Proposed Standard	*SMTP Service Extension for Delivery Status Notifications*, Keith Moore (University of Tennessee), January 15, 1996.
RFC 1892	Proposed Standard	Greg Vaudreuil (Corporation for National Research Initiatives), *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages*, January 15, 1996.
RFC 3464	Full Standard	K. Moore, G. Vaudreuil, *An Extensible Message Format for Delivery Status Notifications*, January 2003.

Security

The list of documents in Table A-6 describe security protocols.

Table A-6 Security

Standard	Status	Description
RFC 1731	Proposed Standard	John G. Myers, *IMAP4 Authentication Mechanisms*, December 1994.
RFC 2195	Proposed Standard	J. Klensin, R. Catoe, P. Krumviede, *IMAP/POP AUTHorize Extension for Simple Challenge/Response*, September 1997.
RFC 2222	Proposed Standard	John G. Myers, *Simple Authentication and Security Layer (SASL)*, October 1997.
RFC 2246	Proposed Standard	T. Dierks, C. Allen, *The TLS Protocol Version 1.0*, January 1999.
RFC 2505 BCP 30	Best Current Practice	G. Lindberg, *Anti-Spam Recommendations for SMTP MTAs*, February 1999.
RFC 2554	Proposed Standard	John G. Myers, *SMTP Service Extension for Authentication*, March 1999.

Table A-6 Security *(Continued)*

Standard	Status	Description
RFC 2595	Proposed Standard	C. Newman, *Using TLS with IMAP, POP3, and ACAP*, June 1999. (Supported by MMP, POP and IMAP.)
RFC 2831	Proposed Standard	P. Leach, C. Newman, *Using Digest Authentication as a SASL Mechanism, May 2000.* (Not yet supported by MMP.)
RFC 3207	Proposed Standard	P. Hoffman, *SMTP Service Extension for Secure SMTP over Transport Layer Security*, February 2002.

Domain Name Service

The documents listed in Table A-7 specify the naming facilities of the Internet and how those facilities are used in messaging.

Table A-7 Domain Name Service

Standard	Status	Description
RFC 920	Policy	Jonathan B. Postel and Joyce K. Reynolds, USC/Information Sciences Institute, *Domain Requirements*, October 1984.
RFC 974	Standard	Craig Partridge, CSNET CIC BBN Laboratories Inc., *Mail Routing and the Domain System*, January 1986.
RFC 1032	Information	Mary K. Stahl, SRI International, *Domain Administrators Guide*, November 1987.
RFC 1033	Information	Mark K. Lottor, SRI International, *Domain Administrators Operations Guide*, November 1987.
RFC 1034	Standard	Paul V. Mockapetris, USC/Information Sciences Institute, *Domain Names - Concepts and Facilities*, November 1987.
RFC 1035	Standard	Paul V. Mockapetris, USC/Information Sciences Institute, *Domain Names - Implementation and Specification*, November 1987.

Text and Character Set Specifications

The following tables list documents that describe national and international telecommunications and information processing requirements.

NOTE Sun Java System Messaging Server supports additional character set and language standards not listed here.

National and International

Table A-8 contains material pertaining to national and international telecommunications and information exchange standards.

Table A-8 National and International Information Exchange

Standard	Status	Description
IA5	International Standard	ITU-T Recommendation T.50, Fascicle VII.3, Malaga-Torremolinos, *International Alphabet No. 5, International Telecommunication Union*, 1984, Geneva, 1989.
ISO 2022	International Standard	International Organization for Standardization (ISO), *Information processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques*, Ref. No. ISO 2022-1986.
JIS X 0201	National Standard	Japanese Standards Association, *Code For Information Interchange*, JIS X 0201-1976.
JIS X 0208	National Standard	Japanese Standards Association, *Code of the Japanese Graphic Character Set For Information Interchange*, JIS X 0208-1990.
JUNET	Public Network	JUNET Riyou No Tebiki Sakusei Iin Kai (JUNET User's Guide Drafting Committee), *JUNET Riyou No Tebiki (JUNET User's Guide)*, First Edition, February 1988.
printableString ASN.1	International Standard	ITU-T X.680, aligned with ISO/IEC-8824-1 Abstract Syntax Notation One (ASN.1). Appears in LDAP/X.500 attribute data types. Defined jointly by the ISO, ITU-T standards bodies and have been reused in Internet RFCs and ISO, ITU-T standards.
US ASCII	National Standard	American National Standards Institute, ANSI X3.4-1986, *Coded Character Set-7-bit American National Standards Code for information interchange*. New York, 1986.
US LATIN	National Standard	American National Standards Institute, ANSI Z39.47-1985, *Coded Character Set-Extended Latin alphabet code for bibliographic use*. New York, 1985.
UTF-8	International Standard	F. Yergeau, *UTF-8, a transformation format of ISO 10646*, January 1998

Internet References

The documentation in Table A-9 describes Internet communications standards.

Table A-9 Internet References

Standard	Status	Description
RFC 1345	Information	Keld Simonsen, Rationel Almen Planlaegning, Internet Activities Board RFC 1345, *Character Mnemonics & Character Sets*, June 1992.

Table A-9 Internet References *(Continued)*

Standard	Status	Description
RFC 1468	Information	Jun Murai (Keio University), Mark Crispin (University of Washington), *Japanese Character Encoding for Internet Messages*, June 1993.
RFC 1502	Information	Harald Tveit Alvestrand, SINTEF DELAB, Internet Activities Board RFC 1502, *X.400 Use of Extended Character Sets*, August 1993.

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

R

S

T

U

V

W

X