Reference Manual

iPlanet Messaging Server

Release 5.0

806-4818-01
September 2000

Contents

About This Guide

This manual provides reference information about the iPlanet Messaging Server 5.0 product. iPlanet Messaging Server 5.0 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 *iPlanet Messaging Server 5.0 Administrator's Guide*. The administrator's guide describes how to configure, maintain, monitor, and troubleshoot iPlanet Messaging Server 5.0. The reference manual provides information about command-line utilities and configuration files. This information enables you to configure, maintain, monitor, and troubleshoot iPlanet Messaging Server 5.0.

Topics covered in this chapter include:

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

Who Should Read This Book

This manual is intended for highly or moderately technical network administrators with experience in UNIX or NT. These administrators will be configuring, administering, and maintaining iPlanet Messaging Server 5.0. Architects and developers may also use the *iPlanet Messaging Server 5.0 Reference Manual*. 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
- iPlanet Administration Server
- iPlanet Directory Server and LDAP
- Netscape Console

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, "Delegated Administrator Command-line Utilities"

  This chapter describes the utilities for iPlanet Delegated Administrator for Messaging.
- Chapter 4, "Messaging Server Configuration"

  This chapter lists the configuration parameters for the Messaging Server.
- Chapter 5, "MTA Configuration"

  This chapter describes the MTA configuration files.
- Chapter 6, "Messaging Multiplexor"

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

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:

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

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

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

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.

Where to Find Related Information

In addition to this guide, iPlanet Messaging Server 5.0 comes with supplementary information for administrators as well as documentation for end users and developers. Use the following URL to see all the Messaging Server documentation:

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

Listed below are the additional documents that are available:

- iPlanet Messaging Server 5.0 Administrator's Guide
- iPlanet Messaging Server 5.0 Reference Manual
- iPlanet Messaging Server 5.0 Schema Reference
- iPlanet Messaging Server 5.0 Provisioning Guide

Where to Find This Book Online

You can find the iPlanet Messaging Server 5.0 Installation Guide online in PDF and HTML formats. To find this book, use this URL:

```
http://docs.iplanet.com/docs/manuals/messaging/ims50/install/conten
ts.htm
```

Chapter 1

Messaging Server Command-line Utilities

iPlanet Messaging Server 5.0 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." For descriptions of the iPlanet Delegated Administrator for Messaging command-line utilities, see Chapter 3, "Delegated Administrator Command-line Utilities."

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

Table 1-1 Messaging Server Commands

Command	Description	Page
`configutil`	Enables you to list and change Messaging Server configuration parameters.	page 16
`counterutil`	Displays all counters in a counter object. Monitors a counter object.	page 20
`deliver`	Delivers mail directly to the message store accessible by IMAP or POP mail clients.	page 21
`hashdir`	Identifies the directory that contains the message store for a particular account.	page 23
`imsasm`	Handles the saving and recovering of user mailboxes.	page 24
`imsbackup`	Backs up stored messages.	page 27
`imsrestore`	Restores messages from the backup device into the message store.	page 29

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

Command	Description	Page
`imscripter`	The IMAP server protocol scripting tool. Executes a command or sequence of commands.	page 30
`mboxutil`	Lists, creates, deletes, renames, or moves mailboxes (folders).	page 32
`mkbackupdir`	Creates and synchronizes the backup directory with the information in the message store.	page 35
`MoveUser`	Moves a user's account from one messaging server to another.	page 38
`readership`	Reports on how many users other than the mailbox owner have read messages in a shared IMAP folder.	page 41
`reconstruct`	Rebuilds one or more mailboxes, or the master mailbox file, and repairs any inconsistencies.	page 42
`start-msg`	Starts the messaging server processes.	page 44
`stop-msg`	Stops the messaging server processes.	page 44
`stored`	Performs cleanup and expiration operations.	page 45

Command Descriptions

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

configutil

The `configutil` utility enables you to list and change iPlanet Messaging Server 5.0 configuration parameters.

For a list of all configuration parameters, see Chapter 4, "Messaging Server Configuration."

Most iPlanet Messaging Server 5.0 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 and `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.

Location: *server-root*`/bin/msg/admin/bin`

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 `-e`, `-l`, or `-p` *prefix* options.
  - Use `-e` to include configuration parameters with empty values in the list.
  - 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 `-e` option with the `-i` option to import configuration parameters even if the value of the configuration option is empty.
- Include the `-l` option with the `-i` option to import all configuration parameters to the server's local configuration file.

Syntax

```
configutil [-f configdbfile] [command-options] [;language]

configutil -i < inputfile
```

Options

The options for this command are:

Option	Description
`-e`	Lists configuration parameters that do not have values specified. May be used with the `-l`, `-p`, and `-i` options.
`-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. If you use `-e` with `-i`, and specify an option without a value, any existing value for that option is deleted. (If you do not use `-e`, when you specify an option without a value, no change is made to any existing value for that option.) Note that a UNIX command line like `cat` *inputfile* `| configutil -i` is not valid syntax.
`-l` *option*	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.
`-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.

Option	Description
`-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 list all configuration parameters with the prefix `service.imap`, including those with empty values:

```
configutil -e -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.

Location: *server-root*`/bin/msg/admin/bin`

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 counters 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 *server-root*`/msg-`*instance*`/counter/counter`.

Examples

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

```
counter
```

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

```
counterutil -o imapstat -r \
server-root/msg-instance/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.

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

Location on UNIX: *server-root*/bin/msg/store/bin

Syntax

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

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.
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.

Syntax

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

Options

The options for this command are:

Option	Description
`-a`	Appends the directory name to the output.
`-i`	Allows you to use the command in interactive mode.

Examples

```
hashdir user1
```

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.

Syntax

```
imsasm [standard_asm_arguments]
```

Options

The options used in the imsasm utility are also known as standard-asm-arguments.

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.

Option	Description
`-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.

Option	Description
`-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 dog "." 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.

Option	Description
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 *ADM_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 35.

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: *server-root*`/bin/msg/store/bin`

Syntax

```
imsbackup [-a userid] [-b blocking_factor] [-f device]
  [-d datetime] [-i] [-l] [-u file] [-v] path
```

Options

The options for this command are:

Option	Description
`-a` *userid*	Authenticates the specified user.
`-b` *blocking_factor*	Everything written to the backup device is performed by blocks of the size 512x*blocking_factor*. The default is 20.
`-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.
`-f` *device*	Specifies the file name to which the backup is written. If `-f` is not specified, `imsbackup` writes to `stdout`.
`-i`	Ignore links. Used for POP store.
`-l`	Autoloads the backup object.
`-u` *file*	Specifies the object name file to back up.
`-v`	Executes the command in verbose mode.
path	Path or path names to be backed up. You must specify the backup path in one of the following formats: • */msg_store/group/user/mailbox* • */msg_store/group/user* • */msg_store/group* • */msg_store*

Examples

The following example backs up `user1` to `/dev/rmt/0`:

```
imsbackup -f /dev/rmt/0 /mystore/ALL/user1
```

The following example backs up all users under `groupA`

```
imsbackup /mystore/groupA
```

The following example performs a full backup of `mystore`:

```
imsbackup /mystore
```

imsrestore

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

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

Syntax

```
imsrestore [-a userid] [-b blocking_factor] [-c [y | n]]
  [-f device] [-h] [-i] [-m file] [-n] [-t] [-u file]
  [-v] path
```

Options

The options for this command are:

Option	Description
`-a` *userid*	Authenticates the specified user.
`-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]`	Answer yes or no to the question “Do you want to continue?”
`-f` *file*	Specifies the file name from which the backup data will be read. If `-f` is not specified, `imsrestore` reads from `stdin`.
`-h`	Dumps the header.
`-i`	Ignores existing messages. Does not check for existing messages before restore.
`-m` *file*	Specifies the user name mapping file. This is used when renaming user ids.

Option	Description
`-n`	Creates a new mailbox with a `.date` extension (if the mailbox exists). By default, messages are appended to the existing mailbox.
`-t`	Prints a table of contents, but restore is not performed.
`-u` *file*	Specifies the name of the object file to restore.
`-v`	Executes the command in verbose mode.
path	Path or path names to be restored. You must specify the path in one of the following formats: • */msg_store/group/user/mailbox* • */msg_store/group/user* • */msg_store/group* • */msg_store*

Examples

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

```
imsrestore -f backupfile
```

The following example restores the messages for `user1` from the file `backupfile`:

```
imsrestore -f backupfile /mystore/ALL/user1
```

imscripter

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

Requirements: May be run remotely.

Location: *server-root*`/bin/msg/admin/bin`

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: *server_root*`/bin/msg/admin/bin`

Syntax

```
mboxutil [-a] [-c mailbox] [-d mailbox] [-g group]
  [-r oldname newname [partition]] [-l] [-p pattern] [-q domain] [-x]
  [-k mailbox cmd] [-u [userid]]
```

Options

The options for this command are:

Option	Description
`-a`	Lists all user quota information.
`-c` *mailbox*	Creates the specified mailbox.
`-d` *mailbox*	Deletes the specified mailbox.
`-g` *group*	Lists quota information for the specified group.
`-k` *mailbox cmd*	Locks the specified mailbox at the folder level; runs the specified command; after command completes, unlocks the mailbox. When a mailbox is locked the owner can view the messages it contains, but no new messages can be added and no existing messages can be deleted or moved. You should use the `-k` option before performing backups, for example.
`-l`	Lists all of the mailboxes on a server.
`-p` *pattern*	When used in conjunction with the `-l` option, lists only those mailboxes with names that match pattern. You can use IMAP wildcards.
`-q` *domain*	Lists quota information for the specified domain.
`-r` *oldname newname* [*partition*]	Renames the mailbox from *oldname* to *newname*. To move a folder from one partition to another, specify the new partition with the partition option. Note that you cannot rename a user's INBOX. Nor can you use `mboxutil -r` to move mail stored under one user ID to another user ID.
`-u [`*userid*`]`	Lists user information such as current size or message store, quota (if one has been set), and percentage of quota currently in use.
`-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
```

To list usage statistics:

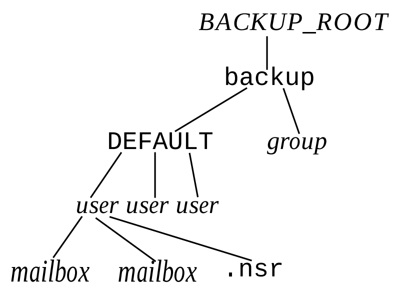
```
mboxutil -u daphne

diskquota size(K) %use msgquota   msgs %use    user
10240     297           no quota   953  29%    daphne
```

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: *server_root*`/bin/msg/store/bin`

The variables in the backup directory contents are:

Variable	Description
BACKUP_ROOT	Message store administrator root directory as specified in the `ims.cnf` file. The default directory is *server_root*`/msg-`*instance*`.`
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 *server_root*`/msg-`*instance*`/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 regex expression specifying user directory names that will go under the *groupname* directory.
user	Name of the message store user.
folder	Name of the user mailbox directory.
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*
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.

Syntax

```
mkbackupdir [-a userid] [-i | -f] [-p directory] [-v]
```

Options

The options for this command are:

Option	Description
`-a` *userid*	Authenticates the specified user.
`-f`	Backs up the folders only. By default, all mailboxes are backed up.
`-i`	Backs up the inbox only. By default, all mailboxes are backed up.

Option	Description
`-p` *directory*	Specifies an alternative directory for the backup image (the default directory is *msg-instance*/`backup`)
`-v`	Executes the command in verbose mode.

Examples

To create the *server_root*/`msg-`*instance*/`backup` directory, enter the following:

```
mkbackupdir
```

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.

Requirements: May be run remotely.

Location: *server-root*/`bin/msg/admin/bin`

Syntax

```
MoveUser -s srcmailhost[:port] -x proxyuser -p password -d destmailhost[:port]
  [-u uid | -u uid -U newuid] [-l ldapURL -D binDN -w password] [options]
```

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 semi-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.
`-s` *srcmailhost*	Source messaging server.
	By default, `MoveUser` assumes IMAP port `143`. To specify a different port, add a semi-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.
`-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=Airius.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://airius.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.airius.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
```

readership

The `readership` utility reports on how many users other than the mailbox owner have read messages in a shared IMAP folder.

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.

The utility scans all mailboxes.

This utility produces one line of output per shared folder, reporting the number of readers followed by a space and the name of the mailbox.

Each reader is a distinct authentication identity that has selected the shared folder within the past specified number of days. Users are not counted as reading their own personal mailboxes. Personal mailboxes are not reported unless there is at least one reader other than the folder's owner.

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

Location: *server-root*`/bin/msg/admin/bin`

Syntax

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

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.

reconstruct

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

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

Location: *server-root*`/bin/msg/admin/bin`

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

Syntax

```
reconstruct [-f] [-p partition] [-r [mailbox [mailbox...]] [-m] [-n]
  [-q] [-o [-d filename]]
```

Options

The options for this command are:

Option	Description
`-f`	Forces `reconstruct` to perform a fix on the mailbox or mailboxes.
`-m`	Performs a high-level consistency check and repair 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`	Do not perform a fix on the mailbox or mailboxes.
`-o`	Checks for orphaned accounts. This option searches for inboxes in the current messaging server host which do not have corresponding entries in LDAP. For example, the `-o` option finds inboxes of owners who have been deleted from LDAP or moved to a different server host. For each orphaned account it finds, `reconstruct` writes the command:
	`mboxutil-d user/`*userid*`/INBOX`
	to the standard output.
`-o -d` *filename*	If `-d` *filename* is specified with the `-o` option, `reconstruct` opens the specified file and writes the `mboxutil -d` commands into that file. The file may then be turned into a script file to delete the orphaned accounts.
`-p` *partition*	Specifies a partition name. You can use this option on the first usage or reconstruct.
`-q`	Fixes any inconsistencies in the quota subsystem, such as mailboxes with the wrong quota root or quota roots with the wrong quota usage reported. The `-q` option can be run while other server processes are running.
`-r` [*mailbox*]	Performs a consistency check and repairs the partition area of the specified mailbox or mailboxes. 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 database.

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`.

start-msg

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

Syntax

```
start-msg [smtp | imap | pop | store | http]
```

Examples

The following command starts all the messaging server processes:

```
start-msg
```

The following command starts the `imap` process:

```
start-msg imap
```

stop-msg

The `stop-msg` utility stops all messaging server processes (`smtp`, `imap`, `pop`, `store`, `http`), or optionally, one specified service.

Syntax

```
stop-msg [smtp | imap | pop | store | http]
```

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
- Cleanup of temporary files on startup
- Implementation of aging policies
- Periodic monitoring of server state, disk space, service response times, and so on
- Issuing of alarms if necessary

The `stored` utility automatically performs cleanup and expiration operations once a day at midnight. You can choose to run additional cleanup and expiration operations.

Requirements: Must be run locally on the Messaging Server.

Location: *server-root*`/bin/msg/admin/bin`

Syntax

To run `stored` from the command line to perform a specific operation:

```
stored [-1] [-c] [-n] [-v [-v]]
```

To run stored as a daemon process:

```
stored [-d] [-v [-v]]
```

Options

The options for this command are:

Option	Description
-c	Performs one cleanup pass to erase expunged messages. Run once, then exits. The -c option is a one-time operation, so you do not need to specify the -1 option.
-d	Run as daemon. Performs system checks and activates alarms, deadlock detection, and database repair.
-1 (the number one)	Run once, then exit.
-n	Run in trial mode only. Does not actually age or cleanup messages. Runs once, then exits.
-v	Verbose output.
-v -v	More verbose output.

Examples

To test expiration policies:

```
stored -n
```

To perform a single aging and cleanup pass:

```
stored -l -v
```

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. These commands are located in the *server_root*`/msg-`*instance*`/` directory.

server-root represents the directory path in which you install the server, and the variable *instance* in `msg-`*instance* represents the server instance you use when you install it (or your host machine name).

The commands are listed in Table 2-1.

Table 2-1 MTA Commands

Command	**Description**	**Page**
`imsimta cache`	Performs operations on the queue cache.	page 49
`imsimta chbuild`	Compiles the MTA character set conversion tables.	page 50
`imsimta cnbuild`	Compiles the MTA configuration files.	page 53
`imsimta convertdb`	Reads the entries in an MTA with version 5.2 or earlier `crdb` database and writes out the entries to a current format MTA `crdb` database.	page 56
`imsimta counters`	Performs operations on the channel counters.	page 57
`imsimta crdb`	Creates an MTA database.	page 59
`imsimta dirsync`	Recreates or updates the MTA directory cache.	page 63
`imsimta find`	Locates the precise filename of the specified version of an MTA log file	page 64
`imsimta kill`	Terminates the specified process.	page 65

Table 2-1 MTA Commands *(Continued)*

Command	Description	Page
`imsimta process`	Lists currently running MTA jobs.	page 66
`imsimta program`	Manipulates the MTA program delivery options.	page 67
`imsimta purge`	Purges MTA log files.	page 69
`imsimta qclean`	Holds or deletes message files containing specific substrings in their envelope From:address, Subject: line, or content.	page 70
`imsimta qm`	Manages MTA message queues.	page 71
`imsimta qtop`	Displays the most frequently occurring envelope From: Subject:, or message content fields found in message files in the channel queues.	page 86
`imsimta refresh`	Combines the functionality of the imsimta cnbuild and imsimta restart utilities.	page 88
`imsimta renamedb`	Renames a MTA database.	page 89
`imsimta restart`	Restarts detached MTA processes.	page 90
`imsimta return`	Returns (bounces) a mail message to its originator.	page 91
`imsimta run`	Processes messages in a specified channel.	page 91
`imsimta start`	Starts the MTA Job Controller and Dispatcher.	page 92
`imsimta stop`	Shuts down the MTA Job Controller and the MTA Dispatcher.	page 93
`imsimta submit`	Processes messages in a specified channel.	page 93
`imsimta test`	Performs tests on mapping tables, wildcard patterns, address rewriting, and URLs.	page 94
`imsimta version`	Prints the MTA version number.	page 103
`imsimta view`	Displays log files.	page 103
`configutil`	Enables you to list and change Messaging Server configuration parameters, including some MTA configuration parameters. See “configutil,” on page 16 for full syntax and description of `configutil`.	page 16

Command Descriptions

You need to be logged in as root (UNIX) or administrator (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 /*server_root*/`msg-`*instance*/`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 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

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` 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, `/`*server_root*`/msg-`*instance*`/imta/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 /*server_root*`/msg-`*instance*`/imta/config/maximum_charset.dat` is read in addition to the file named by the `IMTA_CHARSET_OPTION_FILE` option of the MTA tailor file, `/`*server_root*`/msg-`*instance*`/imta/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-*instance*/imta/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-*instance*/imta/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 `msg-`*instance*`/imta/lib/config_data` by the `IMTA_CONFIG_DATA` option of the MTA tailor file, `msg-`*instance*`/imta/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 itself) 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, the 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-`*instance*`/imta/config/imta.cnf`
- `msg-`*instance*`/imta/config/aliases`
- `msg-`*instance*`/imta/config/mappings`
- `msg-`*instance*`/imta/config/option.dat`
- `msg-`*instance*`/imta/config/conversions`
- `msg-`*instance*`/imta/config/security.cnf`

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-`*instance*`/imta/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`.

Option	Description
-maximum | -nomaximum	msg-*instance*/imta/config/maximum.dat is read in addition to the file named by the IMTA_OPTION_FILE option in the MTA tailor file, msg-*instance*/imta/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_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-*instance*/imta/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-*instance*/imta/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-*instance*/imta/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-*instance*/imta/config/imta_tailor.

Option	Description
`-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 By default, `imsimta cnbuild` is executed whenever the `imsimta refresh`command is invoked.

imsimta convertdb

The format of MTA `crdb` databases has changed with new MTA versions. The `imsimta convertdb` utility reads the entries in an MTA with version 5.2 or earlier `crdb` database and writes out the entries to a current format MTA `crdb` database.

The `imsimta convertdb` utility can also read an MTA 6.0 or later database as input.

Syntax

```
imsimta convertdb input-database-spec output-database-spec
```

Parameters

The parameters for this command are:

Parameter	Description
input-database-spec	The name of the MTA database (usually one created while running an earlier version of the MTA) from which to read entries.
output-database-spec	The name of the MTA version 6.0 or later MTA database to which to write the entries stored in the input MTA database (usually an MTA version 5.2 or earlier database). Special keywords such as `IMTA_ALIAS_DATABASE`, `IMTA_REVERSE_DATABASE`, `IMTA_FORWARD_DATABASE`, `IMTA_GENERAL_DATABASE`, `IMTA_DOMAIN_DATABASE`, and `IMTA_PIPE_DATABASE` are supported; the use of such a special keyword tells the MTA to write the database specified by the corresponding tailor file option.

Examples

The following example converts an MTA for the UNIX alias database to the most current format. The input database, for example, might be an MTA version 5.2 alias database that is being converted to an MTA version 6.0 format.

```
imsimta convertdb aliasesdb.dat IMTA_ALIAS_DATABASE
```

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 [-headers | -noheaders] [-output=file_spec]

imsimta counters -today
```

Options

The options for this command are:

Option	Description
`-clear`	The `-clear` command clears the in-memory channel counters.
`-create`	Creates the in-memory channel counters. Do not use this option if you already have in-memory counters. `imsimta start` creates the in-memory counters. Note that this option should never be used unless you have manually deleted the counters using the `-delete` 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.

Option	Description
`-today`	Counts and displays the number of messages processed on this day. Note that the messages counted are the number of messages processed up until the time that this command is executed.

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.
`-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.

Option	Description
`-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 `-quote`d 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 dirsync

The `imsimta dirsync` utility recreates or updates the MTA directory cache.

This utility is normally run by a `cron` job so there should not be a need to run it manually. `imta dirsync` needs to run any time directory data that affects message delivery changes.

NOTE You must be logged in as `root` in order to run `imimta dirsync`.

Syntax

```
imsimta dirsync [-v] [-l localhost1, localhost2,...] [-F] [-L]
  [-i ldap_filter] [-t]
```

Options

The options for this command are:

Option	Description
`-v`	Runs this command in verbose mode. A trace file is created in the log directory.
`-F`	Performs a full synchronization. By default, the `imsimta dirsync` command performs an incremental synchronization of the directory cache, which means that only entries that have been added or modified in the directory since the last synchronization are updated. The `-F` option causes the directory cache to be completely regenerated, thus creating a faithful image of the directory. The MTA is restarted after a full synchronization.

Option	Description
`-i` *ldap_filter*	Uses the specified filter, instead of the default filter, which is, any entry that has a modifytimestamp or createtimestamp later than the previous dirsync's timestamp.
`-t`	Execute `imsimta dirsync` in the test mode. Searches the directory and prints out the details on invalid entries, if there are any. No changes are made to the cache itself. For details on all entries, test also in verbose mode (run both the `-t` and `-v` options).

Example

To perform a full directory cache synchronization, execute the following command:

```
imsimta dirsync -F
```

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 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 server_root/msg-instance/imsimta/log/tcp_local_slave.log
```

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

```
imsimta find \
server_root/msg-instance/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 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`. A change in an existing one will take effect only after the next full dirsync is performed.

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 locale. 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 (*server-root*`/msg-`*instance*`/imta/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.
`-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.

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 `msg-`*instance*`/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 `msg-`*instance*`/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 | -env_from=substring | -subject=substring]
  [-database] [-delete | -hold] [-directory_tree] [-match=keyword]
  [-min_length=n] [-threads | -nothreads] [-verbose | -noverbose]
  [channel]
```

Options

The options for this command are:

Option	Description
`-content`=*substring* | `-env_from`=*substring* | `-subject`=*substring*	Specifies the substrings for which to search. Any combination of `-content`, `-env_from`, and `-subject` 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`).
`-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.
`-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`.
`-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.

Option	Description
`-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 72 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 | -env_from=substring | -subject=substring]
  [-database | -directory_tree] [-delete | -hold] [-match=keyword]
  [-min_length=n] [-threads=n | -nothreads]
  [-verbose | -noverbose] [channel]
```

The options for this sub-command are:

Option	Description
`-content=`*substring* | `-env_from=`*substring* | `-subject=`*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`).
`-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
`-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
`-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.

counters today

Displays the count of messages processed so far today.

Available for both interactive and non-interactive modes.

```
counters today
```

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 84 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 `msg-`*instance*`/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`	Selects the queue cache database as the source of message information.
`-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`.
`-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 `msg-`*instance*`/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 84 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 84 for information on the `-channel`, `-all`, `-confirm`, and `-log` options.

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 84 for information on using the `-channel`, `-all`, and `-confirm` options.

release

Unmarks as held the specified messages from the most recently generated message queue listing and releases a processing job to process.

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 84 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 84 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.

summarize

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

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

The options for this sub-command are:

Option	Description
`-database` | `-directory_tree`	Controls whether the information presented is gathered from the queue cache database (`-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.

Option	Description
`-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] | -env_from[=range] | -subject[=range]
  [-database | -directory_tree]   [-min_count=n]
  [-threads=n | -nothreads] [-top=n] [-verbose | -noverbose]
  [channel]
```

The options for this sub-command are:

Option	Description
`-content`[=*range*] | `-env_from`[=*range*] | `-subject`[=*range*]	The `-content`, `-env_from`, and `-subject` options are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Use `-env_from` to display frequent envelope From: fields or `-content` to display frequent message contents. Any combination of `-content`, `-env_from`, and `-subject` may be specified. However, only one of each may be used. The `-content`, `-env_from`, 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)`, `-env_from=(START=1,LENGTH=2147483647)`, and `-subject=(START=1,LENGTH=2147483647)`. 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.
`-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`.
`-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 `msg-`*instance*`/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.
`-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.

Option	Description
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: Subject:, or message content fields found in message files in the channel queues.

Syntax

```
imsimta qtop [-content=offset | -env_from=offset | -subject=offset]
  [-database | -directory_tree] [-min_count=n]
  [-threads=n | -nothreads] [-top=n] [-verbose | -noverbose]
  [channel]
```

Options

The options for this command are:

Option	Description
`-content=`*offset* | `-env_from=`*offset* | `-subject=`*offset*	Specifies which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Specify `-env_from` to display frequent envelope From: fields or `-content` to display frequent message contents. 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)`, `-env_from=(START=1, LENGTH=2147483647)`, and `-subject=(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.
`-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`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`

Option	Description
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 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-`*instance*`/imta/db/aliasesdb`).

```
imsimta renamedb tmpdb IMTA_ALIAS_DATABASE
```

imsimta restart

The `imsimta restart` command stops any MTA Job Controller or MTA Service Dispatcher jobs that are running, and restarts the MTA Job Controller and MTA Service Dispatcher.

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]
```

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 jobs:

```
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-`*instance*`/imta/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.

Syntax

```
imsimta run channel [poll]
```

Parameters

The parameters for this command are:

Parameter	Description
channel	Specifies the channel to be processed. This parameter is mandatory.
`poll`	If `poll` is specified, the channel program runs even when there are no messages queued to the channel for processing.

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 multithreaded 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.

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

Syntax

```
imsimta stop [dispatcher | job_controller]
```

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 indictates 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 [address] [-alias_file=filename]
  [-channel | -nochannel]
  [-check_expansions | -nocheck_expansions]
  [-configuration_file=filename ]  [-database=database_list]
  [-debug | -nodebug]  [-delivery_receipt | -nodelivery_receipt]
  [-destination_channel=channel] [-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]
```

```
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]
```

```
imsimta test -match
```

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

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.

Option	Description
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-`*instance*`/imta/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.
`-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-`*instance*`/imta/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.

Option	Description
`-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.
`-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.
`-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-`*instance*`/imta/config/imta_tailor`, which is usually `msg-`*instance*`/imta/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 to `imsimta test -rewrite`. By default, `imsimta test -rewrite` 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.

Option	Description
`-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-`*instance*`/imta/config/imta_tailor`, which is usually the file named `msg-`*instance*`/imta/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.
`-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-`*instance*`/imta/config/imta_tailor`, which is usually the file `msg-`*instance*`/imta/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 -rewrite`. By default, `imsimta test -rewrite` writes output to stout.
`-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-instance/imta/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.

Option	Description
`-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

Delegated Administrator Command-line Utilities

The command-line utilities for iPlanet Delegated Administrator for Messaging manage domain administrators, users, and groups for iPlanet Messaging Server 5.0.

The commands are listed in Table 3-1.

Table 3-1 Delegated Administrator Command Line Interfaces

Command	**Description**	**Which administrator has permission to execute this command**	**Page**
`imadmin admin add`	Grants domain administrator privileges to a user.	Top-level Admin.	page 109
`imadmin admin remove`	Revokes domain administrator privileges from a user.	Top-level Admin.	page 110
`imadmin admin search`	Searches and displays users who have domain administrator privileges.	Anybody	page 112
`imadmin domain create`	Creates a domain.	Top-level Admin.	page 113
`imadmin domain delete`	Deletes a domain.	Top-level Admin.	page 115
`imadmin domain modify`	Modifies a domain.	Top-level Admin.	page 116
`imadmin domain purge`	Purges a domain.	Top-level Admin.	page 118
`imadmin domain search`	Searches for a domain.	Anybody	page 120
`imadmin family create`	Creates a family group.	Top-level, Domain Admins	page 121
`imadmin family delete`	Deletes a family group.	Top-level, Domain Admins.	page 123

Table 3-1 Delegated Administrator Command Line Interfaces *(Continued)*

Command	Description	Which administrator has permission to execute this command	Page
`imadmin family modify`	Modifies a family group.	Top-level, Domain Admins.	page 124
`imadmin family purge`	Purges a family group.	Top-level Admin.	page 126
`imadmin family search`	Searches for a family group.	Anybody	page 128
`imadmin family-admin add`	Grants family administrator privileges to a user.	Top-level, Domain, Family Admins.	page 129
`imadmin family-admin remove`	Revokes family administrator privileges from a user.	Top-level, Domain, Family Admins.	page 131
`imadmin family-admin search`	Searches and displays users who have family administrator privileges.	Anybody	page 132
`imadmin family-member create`	Adds a member to a family group.	Top-level, Domain, Family Admins.	page 134
`imadmin family-member delete`	Marks a family group member for deletion from the directory.	Top-level, Domain, Family Admins.	page 136
`imadmin family-member remove`	Removes the membership of the specified user.	Top-level, Domain, Family Admins.	page 137
`imadmin family-member search`	Searches for a family group member.	Anybody	page 139
`imadmin group create`	Creates a group.	Top-level, Domain Admins and Mail list owner.	page 140
`imadmin group delete`	Deletes a group.	Top-level, Domain Admins and Mail list owner.	page 142
`imadmin group modify`	Modifies a group.	Top-level, Domain Admins and Mail list owner.	page 143
`imadmin group purge`	Purges a group.	Top-level Admin.	page 145
`imadmin group search`	Searches for a group.	Anybody	page 147
`imadmin user create`	Creates a user.	Top-level, Domain Admins.	page 149

Table 3-1 Delegated Administrator Command Line Interfaces *(Continued)*

Command	Description	Which administrator has permission to execute this command	Page
`imadmin user delete`	Deletes a user.	Top-level, Domain Admins.	page 151
`imadmin user modify`	Modifies a user.	Top-level, Domain Admins.	page 152
`imadmin user purge`	Purges a user.	Top-level Admin.	page 154
`imadmin user search`	Searches for a user.	Anybody	page 156

Execution Modes

The command line execution has three possible modes:

- Interactive

  `imadmin` *object task*

  The administrator is queried for the remainder of the options and attributes.

- Execute with options specified in a file

  `imadmin` *object task* `-i` *inputfile*

  Analyzes *inputfile* and executes it.

- Immediate or shell execution

  `imadmin` *object task* [*options*]

Command File Format

Options can be specified within a file, using the `-i` option.

Within the file, option names are separated from option values by white space. The option value begins with the first non-white space character and extends to the end-of-line character. Option sets are separated by blank lines.

The general syntax is:

```
<option name><white space>[option value, if any]
<option name><white space>[option value, if any]
...
<option name><white space>[option value, if any]
<blank line>
<option name><white space>[option value, if any]
<option name><white space>[option value, if any]
...
<option name><white space>[option value, if any]
```

The command line values become the default for each option set. Alternatively, these options can be specified for each option set. The value then overrides any default specified on the command line.

The following shows an example of the format and syntax for the file specified by the `-i` option for the `imadmin user add` command.

```
l newuser1
F new
L user1
W secret

l newuser2
F new
L user2
W secret

l newuser3
F new
L user3
W secret

<and so on...>
```

Command Descriptions

This section provides descriptions, syntax, and examples for the Delegated Administrator commands.

imadmin admin add

The `imadmin admin add` command adds domain administrators for a particular domain.

The `imadmin admin add` command can also be used to grant Domain Administrator privileges to a user.

Syntax

```
imadmin admin add -D login -l login -n domain -w password
  [-d domain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the Top-level Administrator.
`-l` *login*	The uid of the user to whom you want to grant administrative privileges.The user should be present in the directory.
`-n` *domain*	The domain of the Top-level Administrator.
`-w` *password*	The password of the Top-level Administrator.

The following options are non-mandatory:

Options	Description
`-d` *domain*	The domain to which you want to grant administrative privileges. If not specified, the domain specified by the `-n` option is used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.

Options	Description
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Examples

The following grants domain administrator privileges to the user with userid `admin1`.

```
imadmin admin add -D chris -n siroe.com -w bolton -l admin1
```

The following grants domain administrator privileges to the user with userid `admin2` for the domain `acme2.com`.

```
imadmin add admin -D chris -w bolton -l admin2 -n acme2.com
```

imadmin admin remove

The `imadmin admin remove` command removes domain administrator privileges from a user. To remove domain administrator privileges from multiple users, use the `-i` option.

Syntax

```
imadmin admin remove -D login -l userid -n domain -w password
  [-d domain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the Top-level Administrator.
`-l` *userid*	The user id of the user to whom administrator privileges are revoked.
`-n` *domain*	The domain of the Top-level Administrator.
`-w` *password*	The password of the Top-level Administrator.

The following options are non-mandatory:

Option	Description
`-d` *domain*	The domain to which administrator privileges are revoked.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

The following command removes domain administrator privileges from the administrator with user id `admin5`:

```
imadmin admin remove -D chris -n siroe.com -w bolton \
-l admin5 -d test.com
```

imadmin admin search

The `imadmin admin search` command searches and displays users who have domain administrator privileges.

Syntax

```
imadmin admin search -D login -n domain -w password
  [-d domain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the -D option.

The following options are non-mandatory:

Option	Description
`-d` *domain*	Searches for users who have domain administrator privileges for the specified domain. If `-d` is not specified, the domain specified by `-n` is used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.

Option	Description
-X *idahost*	Specifies an alternate host on which the enterprise server is running. If the -X option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
-s	Use SSL (Secure Socket Layer) to connect to the iDA server.
-v	Enable debugging output.

Example

To search for all domain administrators of the `test.com` domain:

```
imadmin admin search -D chris -n siroe.com -w bolton \
-d test.com
```

imadmin domain create

The `imadmin domain create` command creates a single domain in the iMS 5.0 system. To create multiple domains, use the `-i` option.

Syntax

```
imadmin domain create -D login -d domain -H mailhost -n domain
  -w password [-A [+|-]attributename:value] [-h] [-i inputfile] [-p idaport]
  [-t domaincontainer] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
-D *login*	The user id of the Top-level Administrator.
-d *domain*	The name of the domain that is being created.

Option	Description
`-H` *mailhost*	The mail host to which this domain responds (for example, `mailhost.bavo.com`).
`-n` *domain*	The domain of the Top-level Administrator.
`-w` *password*	The password of the Top-level Administrator.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-t` *domaincontainer*	The domain container DN for the domain. This is the pointer into the tree where the domain's users and groups are stored. If this option is not specified then a domain container is created under the `osisuffix` specified in the iDA servlet properties.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To create a new domain, enter:

```
imadmin domain create -D chris -d test.com -n siroe.com \
-w bolton
```

imadmin domain delete

The `imadmin domain delete` command deletes a single hosted domain from the iMS 5.0 system and sets `inetdomainstatus` to “delete.” To delete multiple hosted domains, use the `-i` option.

No undelete utility exists. However, the administrator can use the `ldapmodify` command to change the status attribute of a domain entry to active at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin domain delete -D login -d domain -n domain -w password [-h]
  [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	**Description**
`-D` *login*	The user id of the Top-level Administrator.
`-d` *domain*	The domain that is being deleted.
`-n` *domain*	The domain of the Top-level Administrator.
`-w` *password*	The password of the Top-level Administrator.

The following options are non-mandatory:

Option	Description
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To delete an existing domain:

```
imadmin domain delete -D chris -d test.com -n siroe.com \
-w bolton
```

imadmin domain modify

The `imadmin domain modify` command modifies attributes of a single domain's directory entry. To modify multiple domains, use the `-i` option.

Syntax

```
imadmin domain modify -D login -d domain -n domain -w password
  [-A [+|-]attributename:value] [-h] [-i inputfile] [-p idaport] [-X idahost]
  [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the Top-level Administrator.
`-d` *domain*	The domain to be modified.
`-n` *domain*	The domain of the Top-level Administrator.
`-w` *password*	The password of the Top-level Administrator.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To modify an existing domain:

```
imadmin domain modify -D chris -w bolton -n siroe.com \
-d domain1.com -A mailhosts:test.sun.com
```

imadmin domain purge

The `imadmin domain purge` command permanently removes all deleted domains from the iMS 5.0 system.

As part of periodic maintenance operations, use the `imadmin domain purge` command to remove all domains that have been deleted for a time period that is longer than the specified grace period.

You can perform a purge at any time by invoking the command manually.

When you invoke the command, these actions occur in the following order:

1. The directory is searched and a list of iMS 5.0 domains is created whose entries include domains that have been marked for deletion longer than the specified grace period. (The default value for the grace period is initially set to 10 days at the time of installation.)
2. Each domain's entire directory entry is removed if the `inetdomainstatus` attribute is deleted. Each domain is stripped of mail related attributes if the `maildomainstatus` attribute is deleted.
3. All users and mail lists within each domain are also removed or stripped.

No undelete utility exists. However, the administrator can use the `ldapmodify` command to change the status attribute of a domain entry to active at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin domain purge -D login -d domain -n domain -w password
  [-g grace] [-h] [-i inputfile] [-P] [-p idaport] [-r] [-X idahost]
  [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the Top-level Administrator.
`-d` *domain*	The domain to be purged.
`-n` *domain*	Domain of the Top-level Administrator.
`-w` *password*	Password of the Top-level Administrator.

The following options are non-mandatory:

Option	Description
`-g` *grace*	Grace period in days before the domain is purged. Domains marked for deletion for less than *grace* days will not be purged. A 0 indicates purge immediately. The default value is read from the configuration file on the server. At installation time the default value is set to 10 days.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-r`	Removes the entire subtree rooted at the domain entry's node.
`-P`	Preview only. Does not perform the purge.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To purge an existing domain:

```
imadmin domain purge -D chris -d test.com -n siroe.com \
-w bolton
```

imadmin domain search

The `imadmin domain search` command obtains all the directory properties associated with a single domain. To obtain all the directory properties for multiple domains, use the `-i` option.

Syntax

```
imadmin domain search -D domain -n domain -w password
   [-d domain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *domain*	Search for this domain.
`-h`	Prints command usage syntax.
`-i` *inputfield*	Reads the command information from a file instead of from the command line.

Option	Description
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

imadmin family create

The `imadmin family create` command creates a single family group in the iMS 5.0 system. To add multiple family groups, use the `-i` option.

Syntax

```
imadmin family create -D login -m familyname -n domain -u userid
  -w password [-A [+|-]attributename:value] [-d familydomain] [-h]
  [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-m` *familyname*	The name of the family group. *familyname* must be a single word without any spaces.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-u` *userid*	The userid of the person to whom billing information is sent.

Option	Description
`-w` *password*	The password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A “+” before the *attributename* indicates adding the value to the current list of attributes. A “-” indicates removing the value.
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To create a new family group, `smith`, enter:

```
imadmin family create -D chris -n siroe.com -w secret \
-m smith -u john
```

imadmin family delete

The `imadmin family delete` command deletes a single family group from the iMS 5.0 system and sets the `mnggrpstatus` to “deleted.” To delete multiple family groups, use the `-i` option.

Members of the family group are deleted when a family group is deleted.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a family group entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin family delete -D login -m familyname -n domain -w password
  [-d familydomain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with the permission to execute this command.
`-m` *familyname*	The name of the family group. *familyname* must be a single word without any spaces.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.

Option	Description
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the directory server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To delete an existing family group:

```
imadmin family delete -D chris -n siroe.com -w bolton -w smith
```

imadmin family modify

The `imadmin family modify` command modifies attributes of a single family group's directory entry. To modify multiple family groups, use the `-i` option.

Syntax

```
imadmin family modify -D login -m familyname -n domain -w password
  [-A [+|-]attributename:value] [-d familydomain] [-h] [-i inputfile]
  [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-m` *familyname*	The name of the family group. *familyname* must be a single word without any spaces.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	The password of user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To modify an existing family group:

```
imadmin family modify -D chris -m smith -n siroe.com \
-w bolton -A description:”new family”
```

imadmin family purge

The `imadmin family purge` command permanently removes all deleted family groups from the iMS 5.0 system.

As part of periodic maintenance operations, use the `imadmin family purge` command to remove all family groups that have been deleted for a time period that is longer than the specified grace period.

You can perform a purge at any time by invoking the command manually.

When you invoke the command, the following actions occur:

1. The directory is searched and a list of iMS 5.0 family groups is created whose entries include family groups that have been marked for deletion longer than the specified grace period. (The default value for the grace period is initially set to 10 days at the time of installation.)
2. Each family group’s entire directory entry is removed.
3. All the users in the family group are also purged when the family group is purged.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a family group entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin family purge -D login -m familyname -n domain -w password
  [-d familydomain] [-g grace] [-h] [-i inputfield] [-P] [-p idaport]
  [-X idahost] [-s] [-v
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-m` *familyname*	The name of the family group. *familyname* must be a single word without any spaces.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	The domain of the family group to be purged.
`-g` *grace*	The grace period in days before the family group is purged. Family groups marked for deletion for less than *grace* days will not be purged. A 0 indicates purge immediately. The default value is read from the configuration file on the server. At installation time the default value is set to 10 days.
`-h`	Prints command usage syntax.
`-i` *inputfield*	Reads the command information from a file instead of from the command line.
`-P`	Preview only, without performing any action.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To purge an existing family group:

```
imadmin family purge -D chris -n siroe.com -w bolton \
-d domain.com -m familyname
```

imadmin family search

The `imadmin family search` command obtains all the directory properties associated with a single family group. To obtain all the directory properties for multiple family groups, use the `-i` option.

Syntax

```
imadmin family search -D login -n domain -w password
  [-d familydomain] [-h] [-i inputfile] [-m familyname] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	The domain of the family group
`-h`	Prints command usage syntax.

Option	Description
`-i` *inputfield*	Reads the command information from a file instead of from the command line.
-m *familyname*	Name of the family group.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

The following example searches for family groups in the `domain1.com` domain:

```
imadmin family search -D chris -w bolton -d domain1.com \
-n siroe.com
```

imadmin family-admin add

The `imadmin family-admin add` command grants a user family administrator privileges.

Syntax

```
imadmin family-admin add -D login -l login -m familyname -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *login*	User id of the family administrator.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To grant family administrator privileges to a user with userid `parent1` to the family group `Smith`:

```
imadmin family-admin add -D chris -n siroe.com -w bolton \
-d test1.com -l parent1 -m Smith
```

imadmin family-admin remove

The `imadmin family-admin remove` command revokes Family Administrator privileges from a user.

Syntax

```
imadmin family-admin remove -D login -l login -m familyname -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport] [-X idahost]
  [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *login*	User id of the family administrator.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To remove family administrator privileges to a user with userid `parent1` to the family group `Smith`:

```
imadmin family-admin remove -D chris -n siroe.com -w bolton \
-d test1.com -l parent1 -m Smith
```

imadmin family-admin search

The `imadmin family-admin` search command searches for and displays users who have Family Administrator privileges for a particular family group.

Syntax

```
imadmin family-admin search -D login -m familyname -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

```
imadmin family-admin search -D chris -w bolton -n siroe.com \
-m MyFamily
```

imadmin family-member create

The `imadmin family-member create` command adds a user to a particular family group.

Syntax

```
imadmin family-member create -D login -F firstname -H mailhost
  -L lastname -l login -m familyname -n domain -w password -W password
  [-A [+|-]attributename:value] [-d familydomain] [-h] [-I initial]
  [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-F` *firstname*	The first name of the family member.
`-H` *mailhost*	Family member's mail host.
`-L` *lastname*	Last name of the family member.
`-l` *login*	User id of the family member.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

`-W` *password*	Password of the family member.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A “+” before the *attributename* indicates adding the value to the current list of attributes. A “-” indicates removing the value.
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-I` *initial*	Middle initial of the family member.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To create a family member with userid `peter` to the family group `Athens4`:

```
imadmin family-member create -D chris -n siroe.com -w bolton \
-d test.com -l peter -m Athens4 -F Peter -L Beck -W secret
```

imadmin family-member delete

The `imadmin family-member delete` command marks a family group member as deleted. To remove the entry from the directory, use the `imadmin user purge` command.

Syntax

```
imadmin family-member delete -D login -l login -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	**Description**
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *login*	User id of the family member.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	**Description**
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.

Option	Description
`-d` *familydomain*	Domain of the family group.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To mark a family member with userid `bill` as deleted from the family group `Athens4`:

```
imadmin family-member delete -D chris -n siroe.com -w bolton \
-l bill -m Athens4
```

imadmin family-member remove

The `imadmin family-member remove` command removes the membership of the specified user.

Syntax

```
imadmin family-member remove -D login -l login -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *login*	User id of the family member.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To remove a family member, execute:

```
imadmin family-member remove -D chris -n siroe.com -w bolton \
-d test.com -l john -m Family1
```

imadmin family-member search

The `imadmin family-member search` command searches for a member of a family group.

Syntax

```
imadmin family-member search -D login -m familyname -n domain
  -w password [-d familydomain] [-h] [-i inputfile] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with the permission to execute this command.
`-m` *familyname*	Name of the family group.
`-n` *domain*	Domain of the user specified with the `-D` option.
`-w` *password*	Password of the user specified with the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *familydomain*	Domain of the family group.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.

Option	Description
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

imadmin group create

The `imadmin group create` command adds a single group to the iMS 5.0 system. To create multiple groups, use the `-i` option.

An email distribution list is one type of group. When a message is sent to the group address, iMS 5.0 sends the message to all members in the group.

Syntax

```
imadmin group create -D login -G groupname -n domain -w password
  [-A [+|-]attributename:value] [-d groupdomain] [-h] [-H mailhost]
  [-i inputfile] [-m user] [-o owner] [-p idaport] [-r moderator]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user who has permission to execute this command.
`-n` *domain*	The domain of the user specified by the `-D` option.
`-G` *groupname*	The name of the group (for example, `mktg-list`).
`-w` *password*	The password of the user specified by the `-D` option.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-d` *groupdomain*	The fully qualified domain name (for example, `bravo.com`). The default is the local domain.
`-h`	Prints command usage syntax.
`-H` *mailhost*	The mail host to which this group responds (for example, `mailhost.bavo.com`). The default is the local mail host.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-m` *user*	User id of the members. If more than one member, use multiple `-m` options.
`-o` *owner*	The group owner's email address. An owner is the individual responsible for the distribution list. An owner can add or delete distribution list members.
`-r` *moderator*	The moderator's email address.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To create a group `testgroup` to the domain `domain1.com`:

```
imadmin group create -D chris -n siroe.com -w bolton \
-G testgroup -d domain1.com
```

imadmin group delete

The `imadmin group delete` command deletes a single group from the iMS 5.0 system. To delete multiple groups, use the `-i` option.

When you invoke the imadmin group delete command, the `inetmailgroupstatus` attribute of the group is set to deleted.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a group entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin group delete -D login -G groupname -n domain -w password
  [-d groupdomain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following are mandatory options:

Option	Description
`-D` *login*	The user id of the user who has permission to execute this command.
`-G` *groupname*	The name of the group to be deleted. For example, `mktg-list`.
`-n` *admindomain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following are non-mandatory options:

Option	Description
`-d` *groupdomain*	The domain of the group. If `-d` is not specified, the domain specified by the `-n` option is used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To delete the group `testgroup@domain1.com`:

```
imadmin group delete -D chris -G testgroup@domain1.com \
-n siroe.com -w bolton
```

imadmin group modify

The `imadmin group modify` command changes the attributes of a single group that already exists in the iMS 5.0 system. To change multiple groups, use the `-i` option.

A mailing list is one type of group. When a message is sent to the group address, iMS 5.0 sends the message to all members in the group.

Syntax

```
imadmin group modify -D login -G groupname -n domain -w password
  [-A [+|-]attributename:value] [-d groupdomain] [-h] [-i inputfile]
  [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following are mandatory options:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-G` *groupname*	The name of the group to be modified. For example, `mktg-list`. The name of the group cannot be modified.
`-n` *admindomain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following are non-mandatory options:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-d` *groupdomain*	The domain of the group. If `-d` is not specified, the domain specified by the `-n` option is used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.

Option	Description
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To modify the group `testgroup@domain1.com`:

```
imadmin group modify -D chris -G testgroup@domain1.com \
-n siroe.com -w bolton
```

imadmin group purge

The `imadmin group purge` command permanently removes all deleted groups from the iMS 5.0 system.

As part of periodic maintenance operations, use the `imadmin group purge` command to permanently remove all groups that have been deleted for a time period that is longer than the specified grace period.

You can perform a purge at any time by invoking the command manually.

When you invoke the command, the following actions occur:

1. The directory is searched and a list of iMS 5.0 groups is created whose entries include groups that have been marked for deletion longer than the specified grace period. (The default value for the grace period is initially set to 10 days at the time of installation.)
2. Each group’s entire directory entry is removed or stripped of all mail related attributes if the `-S` option is specified.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a group entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin group purge -D login -G groupname -n domain -w password
  [-d groupdomain] [-g grace] [-h] [-i inputfield] [-P] [-p idaport]
  [-S] [-s] [-v] [-X idahost]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-G` *groupname*	The name of the group to be modified. For example, `mktg-list`. The name of the group cannot be modified.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the -D option.

The following options are non-mandatory:

Option	Description
`-d` *groupdomain*	The domain of the group to be purged. If `-d` is not specified, the domain of `-n` is used.
`-g` *grace*	The grace period in days before the group is purged. Groups marked for deletion for less than *grace* days will not be purged. A 0 indicates purge immediately. The default value is read from the configuration file on the server. At installation time the default value is set to 10 days.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-P`	Preview only.

Option	Description
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-S`	Strip mail attributes only.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To purge an existing group:

```
imadmin group purge -D chris -n siroe.com -w bolton \
-G groupname
```

imadmin group search

The `imadmin group search` command obtains all the directory properties associated with a single group. To obtain all the directory properties for multiple groups, use the `-i` option.

Syntax

```
imadmin group search -D login -n domain -w password [-d groupdomain]
  [-G groupname] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-n` *domain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *groupdomain*	The domain of the group to be searched. If `-d` is not specified, the domain of `-n` is used.
`-G` *groupname*	The name of the group to be searched. For example, `mktg-list`. The name of the group cannot be modified.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To search new groups:

```
imadmin group search -D chris -n siroe.com -w password \
-G=newgroup
```

imadmin user create

The `imadmin user create` command creates a single user to the iMS 5.0 system. To create multiple users, use the `-i` option.

Syntax

```
imadmin user create -D login -F firstname -L lastname -l userid
  -n domain -W password -w password [-A [+|-]attributename:value]
  [-d userdomain] [-H hostname] [-h] [-I initial] [-i inputfile]
  [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-F` *firstname*	The user's first name.
`-L` *lastname*	The user's last name.
`-l` *userid*	The user's login name.
`-n` *domain*	The domain of the user specified by the `-D` option.
`-W` *password*	The user's password.
`-w` *password*	The password of the user specified by the `-D` option.

The following options are non-mandatory:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-d` *userdomain*	The domain of the user. If `-d` is not specified, the value of `-n` is used.
`-H` *mailhost*	The mail host to which this user responds (for example, `mailhost.bavo.com`). The default is the local mail host.
`-h`	Prints command usage syntax.
`-I` *initial*	The user's middle initial.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

The following command creates a user:

```
imadmin user create -D chris -n siroe.com -w bolton -F Rachel \
-L Smith -l rsmith -W secret
```

imadmin user delete

The `imadmin user delete` command deletes a single user from the iMS 5.0 system and sets the `inetuserstatus` to "deleted." To delete multiple users, use the `-i` option.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a user entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin user delete -D login -l username -n domain -w password
  [-d userdomain] [-h] [-i inputfile] [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *username*	The user's user id.
`-n` *admindomain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *userdomain*	The domain of the user. If `-d` is not specified, the domain of `-n` is assumed.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.

Option	Description
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To delete a user:

```
imadmin user delete -D chris -l user1 -n siroe.com -w bolton
```

imadmin user modify

The `imadmin user modify` command changes the attributes of a single user that already exists in the iMS 5.0 system. To change multiple users, use the `-i` option.

Syntax

```
imadmin user modify -D login -l userid -n domain -w password
  [-A [+|-]attributename:value] [-d userdomain] [-h] [-i inputfile]
  [-p idaport] [-X idahost] [-s] [-v]
```

Options

The following are mandatory options:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-l` *userid*	The userid of the user to be modified.

Option	Description
`-n` *domain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following are non-mandatory options:

Option	Description
`-A` [+ | -]*attributename*:*value*	An attribute to modify. The *attributename* is defined in the LDAP schema and value replaces any and all current values for this attribute in the directory. You can repeat this option to modify multiple attributes at the same time, or to specify multiple values for the same attribute. A "+" before the *attributename* indicates adding the value to the current list of attributes. A "-" indicates removing the value.
`-d` *userdomain*	The domain of the user. If `-d` is not specified, the domain specified by the `-n` option is used.
`-h`	Prints command usage syntax.
`-i` *inputfile*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To modify the user `user1@domain1.com`:

```
imadmin user modify -D chris -l user1@domain1.com \
-n siroe.com -w bolton
```

imadmin user purge

The `imadmin user purge` command permanently deletes a single user from the iMS 5.0 system. To permanently delete multiple users, use the `-i` option.

As part of periodic maintenance operations, use the `imadmin user purge` command to permanently delete all users that have been deleted by the status attribute for a time period that is longer than the specified grace period.

You can perform a purge at any time by invoking the command manually.

When you invoke the command, the following actions occur:

1. The directory is searched and a list of iMS 5.0 users is created whose entries include users that have been marked for deletion longer than the specified grace period. (The default value for the grace period is initially set to 10 days at the time of installation.)
2. The `mboxutil` utility is invoked to delete each user's store mailbox.
3. Each user's entire directory entry is removed if the `inetuserstatus` is deleted. Each user is stripped of mail related attributes if the `mailuserstatus` attribute is deleted.

No undelete utility exists. However, you can use the `ldapmodify` command to change the status attribute of a user entry to `active` at any time before the purge grace period has expired and a purge is set to run against the entry.

Syntax

```
imadmin user purge -D login -l userid -n domain -w password
  [-d userdomain] [-g grace] [-h] [-i inputfield] [-P] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
-l *userid*	The user id of the user to be purged.
`-n` *domain*	The domain of the user specified by the `-D` option.
`-w` *password*	The password of the user specified by the `-D` option.

The following options are non-mandatory:

Option	Description
`-d` *userdomain*	The domain of the user to be purged. If `-d` is not specified, the domain of `-n` is used.
`-g` *grace*	The grace period in days before the user is purged. Users marked for deletion for less than *grace* days will not be purged. A 0 indicates purge immediately. The default value is read from the configuration file on the server. At installation time the default value is set to 10 days.
`-h`	Prints command usage syntax.
`-i` *inputfield*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To purge an existing user:

```
imadmin user purge -D chris -n siroe.com -w bolton -l scott
```

imadmin user search

The `imadmin user search` command obtains all the directory properties associated with a single user. To obtain all the directory properties for multiple users, use the `-i` option.

Syntax

```
imadmin user search -D login -n domain -w password [-d userdomain]
  [-F firstname] [-h] [-i inputfile] [-L lastname] [-l userid] [-p idaport]
  [-X idahost] [-s] [-v]
```

Options

The following options are mandatory:

Option	Description
`-D` *login*	The user id of the user with permission to execute this command.
`-n` *domain*	The domain of the user specified with the `-D` option.
`-w` *password*	The password of the user specified with the -D option.

The following options are non-mandatory:

Option	Description
`-F` *firstname*	The user’s first name.
`-L` *lastname*	The user’s last name
`-l` *userid*	The user’s user id.
`-h`	Prints command usage syntax.

Option	Description
`-i` *inputfield*	Reads the command information from a file instead of from the command line.
`-p` *idaport*	Use this option to specify an alternate TCP port where the iDA server is listening. If not specified, the default *idaport* will be used, or 80 if no default was configured at install time.
`-X` *idahost*	Specifies an alternate host on which the enterprise server is running. If the `-X` option is specified and that server does not respond, then the command will fail; it does not try to connect to the default server. If not specified, the default *idahost* will be used, or the localhost if no default was configured at install time.
`-s`	Use SSL (Secure Socket Layer) to connect to the iDA server.
`-v`	Enable debugging output.

Example

To search for a user with the login `testuser`:

```
imadmin user search -D chris -n siroe.com -w bolton \
-l testuser
```

Chapter 4

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 16.

For information about configuring the MTA, see Chapter 5, "MTA Configuration."

configutil Parameters

Table 4-1 configutil Parameters

Parameter	Description
`nsclassname`	
`alarm.msgalarmnoticehost`	
`alarm.msgalarmnoticeport`	Default: 25
`alarm.msgalarmnoticercpt`	Default: Postmaster@localhost
`alarm.msgalarmnoticesender`	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.*.msgalarmdescription`	Alarm description.
`alarm.*.msgalarmstatinterval`	Default: 3600
`alarm.*.msgalarmthreshold`	

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`alarm.*.msgalarmthresholddirection`	Checks for threshold condition. 1 (default) for over, -1 for under.
`alarm.*.msgalarmwarninginterval`	Minimum interval to send duplicate warning (hours). Default: 168
`alarm.diskavail.msgalarmdescription`	Percentage mail partition diskspace available.
`alarm.diskavail.msgalarmstatinterval`	checking interval (seconds). Set to 0 to disable checking of disk usage. Default: 3600.
`alarm.diskavail.msgalarmthreshold`	Default: 10
`alarm.diskavail.msgalarmthresholddirection`	Default: -1
`alarm.diskavail.msgalarmwarninginterval`	Default: 24
`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`	Default: 10
`alarm.serverresponse.msgalarmthresholddirection`	Default: 1
`alarm.serverresponse.msgalarmwarninginterval`	Default: 24
`encryption.nscertfile`	cert file location.
`encryption.nskeyfile`	key file location.
`encryption.nsssl2`	Default: no
`encryption.nsssl2ciphers`	
`encryption.nsssl3`	Default: yes
`encryption.nsssl3ciphers`	
`encryption.nsssl3sessiontimeout`	Default: 0
`encryption.nssslclientauth`	Default: 0
`encryption.nssslsessiontimeout`	Default: 0
`encryption.fortezza.nssslactivation`	Default: off
`encryption.rsa.nssslactivation`	Default: on
`encryption.rsa.nssslpersonalityssl`	Default: Server-Cert

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`encryption.rsa.nsssltoken`	Default: internal
`gen.accounturl`	Location of the server administration resource for end users.
`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.
`gen.sitelanguage`	Default language tag. Default: en.
`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`	
`local.deforg.name`	
`local.enduseradmincred`	Password for end user administrator.
`local.enduseradmindn`	User id for end user administrator.
`local.hostname`	DN of Local hostname.
`local.imta.imta_tailor`	Location of the `imta_tailor` file for this MTA instance.
`local.imta.ldsearchtimeout`	Specifies the LDAP search timeout when searching for users and groups. Default: 0.
`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`	Allows the last alias lookup to be skipped. Instead the recipient address is rewritten to a fixed address.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`local.imta.lookupmaxnbfailed`	Defines when the routing process stops performing unsuccessful LDAP searches (in processes). The default: no limit.
`local.imta.hostnamealiases`	List of hostname aliases. Dirsync uses the hostnames in this list and those listed in `local.hostname` to check if an entry is local.
`local.imta.mailalises`	List of LDAP attributes.
`local.imta.schematag`	Defines the types of LDAP entries that are supported by Dirsync. Default: ims50.
`local.imta.ugfilter`	Sets the LDAP search filter that Dirsync uses when searching for users and groups.
`local.imta.statssamplesize`	Sets whether or not Dirsync displays a report of the number of users and group entries. Default: yes.
`local.imta.reversenabled`	Triggers the generation of the Server Side Rules (SSR) database. Default: yes.
`local.imta.vanityenabled`	Controls whether or not vanity domains are enabled. Setting to yes enables vanity domain. If the variable does not exist, the MTA assumes that vanity domain is enabled. Default: yes.
`local.imta.catchallenabled`	Controls whether or not catchall addresses are enabled. Default: yes
`local.imta.scope`	Prompts dirsync to cache only entries for which the mailhost attribute is the local host.
`local.installeddir`	Full pathname of software installation directory.
`local.instancedir`	Full pathname of server instance directory.
`local.lastconfigfetch`	Last configuration fetch timestamp.
`local.ldapbasedn`	Base DN.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`local.ldapcachefile`	Location of cached configuration.
`local.ldaphost`	LDAP server for SIE.
`local.ldapisiedn`	Installed software DN.
`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.queuedir`	Full pathname of spool directory.
`local.report.reportercmd`	Command to run in order to generate reports. Default: *server_root*`/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.
`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.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`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`	Specifies the character set name.
`local.rfc822header.fixlang`	Specifies the two-letter language ID. This parameter must be used in conjunction with the `fixcharset` parameter.
`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.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.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.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`local service.http.smtpauthuser`	User id for end user AUTH SMTP user.
`local.service.pab.attributelist`	
`local.service.pab.enabled`	Enable or disable PAB feature.
`local.service.pab.ldapbasedn`	Base DN for PAB searches.
`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.
`local.store.snapshotinterval`	Message Store DB snapshot interval. Default: 0.
`local.store.snapshotpath`	Pathname for Message Store DB snapshot storage.
`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.
`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.

Table 4-1 configutil Parameters *(Continued)*

Parameter	**Description**
`local.ugldaphost`	LDAP server for user lookup.
`local.ugldapport`	LDAP port. Default: 389.
`local.ugldapuselocal`	Default: no
`local.ugldapusessl`	Sets whether or not to use SSL to connect to LDAP server. Default: no.
`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.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`logfile.*.buffersize`	Size of log buffers (in bytes). Default: 0. * can be one of the following components: `admin`, `default`, `http`, `imap`, `imta`, `pop`.
`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`.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`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. Default: no. * can be one of the following components: `admin`, `default`, `http`, `imap`, `imta`, `pop`.
`logfiles.admin.alias`	
`logfiles.default.alias`	
`logfiles.http.alias`	
`logfiles.imap.alias`	
`logfiles.imta.alias`	
`logfiles.pop.alias`	
`service.authcachesize`	
`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 complete email address without domains.
`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.
`service.http.domainallowed`	List of domains and / or IP address allowed HTTP access.
`service.http.domainnotallowed`	List of domains and / or IP addresses not allowed HTTP access.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`service.http.enable`	Sets whether or not the server is started automatically. Default: yes.
`service.http.enablesslport`	Sets whether or not the service is started on a sslport. Default: no.
`service.http.fullfromheader`	Sets whether or not to send complete “from” header. Default: no.
`service.http.idletimeout`	Idle timeout (in minutes). Default: 3.
`service.http.ipsecurity`	Sets whether or not to restrict session access to login IP addresses. Default: yes.
`service.http.ldappoolsize`	Number of LDAP connections. Default: 1.
`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`	List of domain and IP addresses allowed to proxy auth.
`service.http.resourcetimeout`	Webmail resource reduction timeout (in seconds). Default: 900.
`service.http.sessiontimeout`	Webmail client session timeout. Default: 7200.
`service.http.smtphost`	SMTP relay host.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`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 disable SSL. Default: yes.
`service.imap.allowanonymouslogin`	Allows anonymous login. Default: no.
`service.imap.banner`	IMAP protocol welcome banner.
`service.imap.connlimits`	Maximum number of connections per IP address.
`service.imap.domainallowed`	List of domains and/or IP addresses allowed IMAP access.
`service.imap.domainnotallowed`	List of domains and/or IP addresses not allowed IMAP access.
`service.imap.enable`	Sets whether or not the server is started automatically. Default: yes.
`service.imap.enablesslport`	Sets whether or not service is started on sslport. Default: no.
`service.imap.idletimeout`	Idle timeout (in seconds). Default: 30.
`service.imap.ldappoolsize`	Number of LDAP connections. Default: 1.
`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.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`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 disabled. Default: yes.
`service.imta.ssrenabled`	
`service.ldapmemcache`	Sets whether to enable or disable LDAP SDK memcache feature. Default: no.
`service.ldapmemcachesize`	Cache size in bytes. Default: 131072.
`service.ldapmemcachettl`	Length of time cache entry lives (in seconds). Default: 30.
`service.ldappoolsize`	Number of LDAP connections. Default: 1.
`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.
`service.pop.connlimits`	Maximum number of connections per IP address.
`service.pop.domainallowed`	List of domains and/or IP addresses allowed POP access.
`service.pop.domainnotallowed`	List of domains and/or IP address not allowed POP access.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`service.pop.enable`	Sets whether or not the server is started automatically. Default: yes.
`service.pop.idletimeout`	Idle timeout (in minutes). Default: 10.
`service.pop.ldappoolsize`	Number of LDAP connections. Default: 1.
`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.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.pop.popminpoll`	Minimum client poll interval in seconds. Default: 0.
`service.pop.port`	POP server port number. Default: 110.
`service.pop.sslusessl`	Sets whether or not to disable SSL. Default: yes.
`service.readtimeout`	Length of time permitted to receive “hello” string when checking for server response time. Default: 10.
`service.sslpasswdfile`	Password for each keyfile.
`store.admins`	Space separated list of user ids with Message Store Administrator privileges.
`store.cleanupage`	Minimum amount of time between expunge and cleanup (in hours). Default: 1.
`store.dbcachesize`	Mailbox list database cache size. Default: 8388608
`store.dbtmpdir`	Mailbox list database temporary directory.

Table 4-1 configutil Parameters *(Continued)*

Parameter	Description
`store.defaultacl`	Default ACL.
`store.defaultmailboxquota`	Default mailbox quota, if not specified in user account. Default: -1 (infinite).
`store.defaultmessagequota`	Default message quota, if not specified in user account. Default: -1 (infinite).
`store.defaultpartition`	Default partition.
`store.diskflushinterval`	
`store.expirerule.*.exclusive`	
`store.expirerule.*.folderpattern`	
`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.
`store.expirerule.*.messagesizedays`	Length of time messagesize message can stay.
`store.expirestart`	
`store.partition.*.path`	Store partition directory path.
`store.partition.primary.path`	
`store.quotaenforcement`	Sets whether to turn quotaenforcement on or off. Default: on.
`store.quotaexceededmsg`	Message to be sent to user when quota exceeds store.quotawarn.
`store.quotaexceededmsginterval`	Interval (in days) to wait before sending another quotaexceededmsg. Default: 7.
`store.quotagraceperiod`	Time (in hours) after a mailbox is over quota, before mail to that mailbox gets rejected. Default: 120.
`store.quotanotification`	sets whether quotanotification is turned on or off. Default: on.

Table 4-1 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`	

Chapter 5

MTA Configuration

The following topics are covered in this chapter:

- imta.cnf File
- Channel Definitions
- Channel Configuration Keywords
- The Alias File
- /var/mail Channel Option File
- SMTP Channel Option Files
- Conversions
- Mapping File
- Option Files
- Tailor File
- Dirsync Option File
- Autoreply Option File
- Job Controller
- Dispatcher

The MTA Configuration Files

This section explains the structure and layout of the MTA configuration files. Some configuration modifications can be done using the command-line interface, as described in Chapter 2, "Message Transfer Agent Command-line Utilities." Modifications not possible through the command line can be done 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 can be 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.

Table 5-1 lists the MTA configuration files with a short description.

Table 5-1 MTA Configuration files

File	**Description**	**Page**
Autoreply Option File	Options used by the `autoreply` program. *server_root*/`msg-`*instance*/`imta/config/autoreply.opt`	274
Alias File (mandatory)	Implements aliases not present in the directory. *server_root*/`msg-`*instance*/`imta/config/aliases`	225
Channel Option Files	Many channels use channel options files to set channel specific options. *server_root*/`msg-`*instance*/`imta/config/`*channel*`_option`	228
Conversion File	Used by conversion channel to control message body part conversions. *server_root*/`msg-`*instance*/`imta/config/conversions`	235
Dirsync Option File (mandatory)	Options used by the `dirsync` program. *server_root*/`msg-`*instance*/`imta/config/dirsync.opt`	273
Dispatcher Configuration File (mandatory)	Configuration file for service dispatcher. *server_root*/`msg-`*instance*/`imta/config/dispatcher.cnf`	279
MTA Configuration File (mandatory)	Used for address rewriting and routing as well as channel definition. *server_root*/`msg-`*instance*/`imta/config/imta.cnf`	177
Mapping File (mandatory)	Repository of mapping tables. *server_root*/`msg-`*instance*/`imta/config/mappings`	241
MTA Option File	File of global MTA options. *server_root*/`msg-`*instance*/`imta/config/option.dat`	257

Table 5-1 MTA Configuration files *(Continued)*

File	Description	Page
MTA Tailor File (mandatory)	File to specify locations. *server_root*/`msg-`*instance*/`imta/config/imta_tailor`	269
Job Controller Config. File (mandatory)	Configuration file used by the Job Controller. *server_root*/`msg-`*instance*/`imta/config/job_controller.cnf`	276

Table 5-2 lists the MTA database files with a short description.

Table 5-2 MTA Database Files

File	Description
Address Reversal Database)	Used to change addresses in outgoing mail. This database is created using the `imsimta dirsync` command and is not editable directly. DO NOT EDIT. *server_root*/`msg-`*instance*/`imta/db/reversedb.db`
Alias Database (mandatory)	Implements aliases, mail forwarding, and mailing lists. Changes should be made to the directory and running `imsimta dirsync`. DO NOT EDIT. *server_root*/`msg-`*instance*/`imta/db/aliasesdb.db`
Domain Database	Used for Storing additional rewriting rules. DO NOT EDIT. *server_root*/`msg-`*instance*/`imta/db/domaindb.db`
General Database	Used with domain rewriting rules or in mapping rules, for site-specific purposes. *server_root*/`msg-`*instance*/`imta/db/generaldb.db`
Profile Database (mandatory)	Database to store program delivery, file delivery, and other special delivery mechanism information. This database is also created from information in the directory during `imsimta dirsync`. DO NOT EDIT. *server_root*/`msg-`*instance*/`imta/db/profiledb.db`

imta.cnf File

The `imta.cnf` file contains the routing and address rewriting configuration. 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.

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 or host table.” Each individual channel definition forms a “channel block,” which defines the channels that the MTA can use and the names associated with each channel. 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 5-3 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 an abstract label used to refer to this channel within the rewrite rules.

For detailed information about channel definitions and channel table keywords, refer to the section “Channel Configuration Keywords,” and to Table 5-3.

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 sections describe keywords and how they control 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 5-3; the keywords shown in **`boldface`** are defaults.

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 will be required from a process in order to enqueue mail to the channel. The `imsimta test -rewrite` utility tells you whether you have any keywords in your configuration file that don't match any known rights list identifier.

Table 5-3 Channel Keywords

Keyword	Usage
`addrsperfile`	Number of addresses per message file.
`addrsperjob`	Number of addresses to be processed by a single job.
`allowetrn`	Honor all ETRN commands.
`allowswitchchannel`	Allow switching to this channel from an `allowswitchchannel` channel.
`authrewrite`	Use SMTP AUTH information in header.
`bangoverpercent`	Group `A!B%C` as `A!(B%C).`
`bidirectional`	Channel is served by both a master and slave program.
`blocketrn`	Do not honor ETRN commands.
`blocklimit`	Maximum number of MTA blocks allowed per message.
`cacheeverything`	Cache all connection information.
`cachefailures`	Cache only connection failure information.
`cachesuccesses`	Cache only connection success information.
`charset7`	Default character set to associate with 7-bit text messages.
`charset8`	Default character set to associate with 8-bit text messages.
`charsetesc`	Default character set to associate with text containing the escape character.
`checkehlo`	Check the SMTP response banner for whether to use `EHLO.`
`commentinc`	Leave comments in message header lines intact.
`commentomit`	Remove comments from message header lines.
`commentstrip`	Remove problematic characters from comment fields in message header lines.
`commenttotal`	Strip comments (material in parentheses) everywhere.
`connectalias`	Does not rewrite addresses upon message dequeue.
`connectcanonical`	Rewrite addresses upon message dequeue.
`copysendpost`	Send copies of failures to the postmaster unless the originator address is blank.
`copywarnpost`	Send copies of warnings to the postmaster unless the originator address is blank.
`daemon`	Specify the name of a gateway through which to route mail.

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
`datefour`	Convert date/time specifications to four-digit years.
`datetwo`	Convert date/time specifications to two-digit years.
`dayofweek`	Include day of week in date and time specifications.
`defaultmx`	Channel determines whether or not to do MX lookups from network.
`deferred`	Honor deferred delivery dates.
`defragment`	Reassemble any MIME-compliant message/partial parts queued to this channel.
`destinationfilter`	Specifies the location of channel filter file that applies to outgoing messages.
`domainetrn`	Tell the MTA to honor only those ETRN commands that specify a domain.
`domainvrfy`	Issue `SMTP VRFY` commands using full address.
`ehlo`	Use EHLO on all initial SMTP connections.
`eightbit`	Channel supports 8-bit characters.
`eightnegotiate`	Channel should negotiate use of eight bit transmission, if possible.
`eightstrict`	Channel should reject messages that contain unnegotiated 8-bit data.
`errsendpost`	Send copies of failures to the postmaster if the originator address is illegal.
`errwarnpost`	Send copies of warnings to the postmaster if the originator address is illegal.
`expandchannel`	Channel in which to perform deferred expansion due to application of `expandlimit`.
`expandlimit`	Process an incoming message "offline" when the number of addressees exceeds this limit.
`exproute`	Explicit routing for this channel's addresses.
`fileinto`	Specify effect on address when a mailbox filter `fileinto` operation is applied.
`filesperjob`	Number of queue entries to be processed by a single job.
`filter`	Specify the location of user filter files.
`forwardcheckdelete`	Effects verification of source IP address.
`forwardchecknone`	No forward lookup is done
`forwardchecktag`	Tell the MTA to do a forward lookup after each reverse lookup.
`headerinc`	Place the message header at the top of the message.
`headerlabelalign`	Align header lines.
`headerlinelength`	Fold long header lines.
`headerread`	Apply header trimming rules from an options file to the message headers upon message enqueue (use with caution).

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
headertrim	Applies header trimming rules from an options file to the message headers (use with caution).
identnone	Disable IDENT lookups; perform IP-to-hostname translation.
identnonelimited	Has the same effect as identnone as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header.
identnonenumeric	Disable IDENT lookups and IP-to-hostname translation.
identnonesymbolic	Disable this IDENT lookup; perform IP to host name translation. Only the host name will be included in the Received: header for the message.
identtcp	Perform IDENT lookups on incoming SMTP connections and IP to host name translation.
identtcplimited	Has the same effect as identtcp as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header.
identtcpnumeric	Perform IDENT lookups on incoming SMTP connections; disable IP to hostname translation.
identtcpsymbolic	Enable IDENT protocol (RFC 1413).
ignoreencoding	Ignore Encoding: header on incoming messages.
immediate	Start delivery immediately after submission for messages of second-class or higher priority.
immnonurgent	Start delivery immediately after submission, even for messages with lower-than-normal priority.
improute	Implicit routing for this channel's addresses.
includefinal	Include final form of address in delivery notifications.
inner	Rewrite inner message headers.
innertrim	Apply header trimming rules from an options file to inner message headers (use with caution).
interpretencoding	Interpret Encoding: header on incoming messages.
lastresort	Specify a last-resort host.
linelength	Message lines exceeding this length limit are wrapped.
linelimit	Maximum number of lines allowed per message.
localvrfy	Issue SMTP VRFY command using local address.
logging	Log message enqueues and dequeues into the log file.
mailfromdnsverify	Setting on an incoming TCP/IP channel causes the MTA to verify that an entry in the DNS exists for the domain used on the SMTP MAIL FROM: command, and to reject the message if no such entry exists.
master	Channel is served only by a master program.
master_debug	Generate debugging output in the channel's master program output.

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
`maxblocks`	Maximum number of MTA blocks per message; longer messages are broken into multiple messages.
`maxheaderaddrs`	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines.
`maxheaderchars`	Maximum number of characters per message header line; longer header lines are broken into multiple header lines.
`maxjobs`	Maximum number of jobs that can be created at one time.
`maxlines`	Maximum number of message lines per message; longer messages are broken into multiple messages.
`maxprocchars`	Specifies maximum length of headers to process.
`maysaslserver`	Cause the SMTP server to permit clients to attempt to use SASL authentication.
`maytls`	SMTP client and server allow TLS use.
`maytlsclient`	SMTP client will attempt TLS use.
`maytlsserver`	SMTP server allows TLS use.
`missingrecipientpolicy`	`Controls handling of messages missing recipient header lines.`
`multiple`	Accept multiple destination hosts in a single message copy.
`mustsaslserver`	Cause the SMTP server to insist that clients use SASL authentication; the SMTP server will not accept messages unless the remote client successfully authenticates.
`musttls`	SMTP client and server insist upon TLS use and will not transfer messages with remote sides that do not support TLS.
`musttlsclient`	SMTP client insists upon TLS use and will not send messages to any remote SMTP server that does not support TLS use.
`musttlsserver`	SMTP server insists upon TLS user and will not accept messages from any remote SMTP client that does not support TLS use.
`mx`	TCP/IP network and software supports MX record lookups.
`nobangoverpercent`	Group `A!B%C` as `(A!B)%C` (default).
`nocache`	Do not cache any connection information.
`nodayofweek`	Remove day of week from date/time specifications.
`nodeferred`	Do not honor deferred delivery dates.
`nodefragment`	Do not perform special processing for message/partial messages.
`nodestinationfilter`	Do not perform channel filtering for outgoing messages.
`noehlo`	Never use the SMTP EHLO command.
`noexproute`	No explicit routing for this channel's addresses.
`nofileinto`	Mailbox filter fileinto operator has no effect.

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
nofilter	Do not perform user mailbox filtering.
noheaderread	Do not apply header trimming rules from option file upon message enqueue.
noheadertrim	Do not apply header trimming rules from options file.
noimproute	No implicit routing for this channel's addresses.
noinner	Do not rewrite inner message headers.
noinnertrim	Do not apply header trimming to inner message headers.
nologging	Do not log message enqueues and dequeues into the log file.
nomailfromdnsverify	The MTA does not verify that an entry in the DNS exists for the domain used.
nomaster_debug	Do not generate debugging output in the channel's master program output.
nomx	TCP/IP network does not support MX lookups.
nonrandommx	Perform MX lookups; does not randomize returned entries of equal precedence.
nonurgentblocklimit	Force messages above this size to wait unconditionally for a periodic job.
noreceivedfor	Do not include Envelope to address in `Received:` header line.
noreceivedfrom	Construct `Received`: header lines without including the original `envelope From`: address.
noremotehost	Use local host's domain name as the default domain name to complete addresses.
norestricted	Do not apply RFC 1137 restricted encoding to addresses.
noreverse	Do not apply reverse database to addresses.
normalblocklimit	Force messages above this size to nonurgent priority.
nosasl	SASL authentication will not be permitted or attempted.
nosaslserver	SASL authentication will not be permitted.
nosendetrn	Do not send an `ETRN` command.
nosendpost	Do not send copies of failures to the postmaster.
noserviceall	Master programs should only process the messages that were queued to process after its inception.
noslave_debug	Do not generate slave debugging output.
nosmtp	Channel does not use SMTP.
nosourcefilter	Do not perform channel filtering for incoming messages.
noswitchchannel	Do not switch to the channel associated with the originating host; does not permit being switched to.
notices	Specifies the amount of time that may elapse before notices are sent and messages returned.
notls	SMTP client and server neither attempt nor allow TLS use.

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
notlsclient	SMTP client does not attempt TLS use when sending messages.
notlsserver	SMTP server does not offer or allow TLS use when receiving messages.
novrfy	Do not issue SMTP VRFY commands.
nowarnpost	Do not send copies of warnings to the postmaster.
nox_env_to	Do not add X-Envelope-to header lines while enqueuing.
period	Specifies periodicity of periodic channel service.
periodic	Channel is serviced only periodically; immediate delivery processing is never done.
personalinc	Leave personal name fields in message header lines intact.
personalomit	Remove personal name fields from message header lines.
personalstrip	Strip problematic characters from personal name fields in message header lines.
pool	Specifies processing pool master channel programs run in.
port	Connect to the specified TCP/IP port.
postheadbody	Both the message’s header and body are sent to the postmaster when a delivery failure occurs.
postheadonly	Only the message’s header is sent to the postmaster when a delivery failure occurs.
randommx	Perform MX lookups; randomizes returned entries with equal precedence.
receivedfor	Includes envelope to address in Received header.
receivedfrom	Include the original envelope From: address when constructing Received: header lines.
remotehost	Use remote host's name as the default domain name to complete addresses.
restricted	Apply RFC 1137 restricted encoding to addresses.
returnenvelope	Control use of blank envelope return addresses.
reverse	Apply reverse database to addresses.
saslswitchchannel	Cause incoming connections to be switched to a specified channel upon a client’s successful use of SASL.
sendpost	Sends copies of failures to the postmaster.
sendetrn	Send an ETRN command, if the remote SMTP server says it supports ETRN.
sensitivity*	Set an upper limit on the sensitivity of messages that can be accepted by a channel.
serviceall	Specifies that the master program should attempt to process all messages queued to the channel each time it runs.

Table 5-3 Channel Keywords *(Continued)*

Keyword	Usage
`sevenbit`	Channel does not support 8-bit characters; 8-bit characters must be encoded.
`silentetrn`	Honor all ETRN commands, but without echoing the name of the channel that the domain matched.
`single`	Only one envelope `To:` address per message copy.
`single_sys`	Each message copy must be for a single destination system.
`slave`	Channel is serviced only by a slave program.
`slave_debug`	Generate slave debug output.
`smtp`	Channel uses SMTP.
`smtp_cr`	Accept CR as an SMTP line terminator.
`smtp_crlf`	Require CRLF as the SMTP line terminator.
`smtp_lf`	Accept LF as an SMTP line terminator.
`sourceroute`	Use source routes in the message envelope; synonymous with 822.
`sourcefilter`	Specify the location of channel filter file for incoming messages.
`subdirs`	Use multiple subdirectories.
`submit`	Marks the channel as a submit-only channel.
`suppressfinal`	Suppress the final address form from notification messages.
`switchchannel`	Switch from the server channel to the channel associated with the originating host.
`threaddepth`	Number of messages per thread.
`tlsswitchchannel`	Switch to specified channel upon successful TLS negotiation.
`unrestricted`	Do not apply RFC 1137 restricted encoding to addresses.
`urgentblocklimit`	Force messages above this size to normal priority.
`usereplyto`	Specifies mapping of `Reply-to` header.
`useresent`	Specifies mapping of Resent- headers for non-RFC 822 environments.
`vrfyallow`	Issue a detailed, informative response.
`vrfydefault`	Provide a detailed, informative response, unless the channel option HIDE_VERIFY=1 has been specified.
`vrfyhide`	Issue only a vague, ambiguous response.
`warnpost`	Send copies of warnings to the postmaster.
`x_env_to`	Add `X-Envelope-to` header lines while enqueuing.

Address Interpretation (bangoverpercent, nobangoverpercent)

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

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

or

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

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

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

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

Routing Information in Addresses (exproute, noexproute, improute, noimproute)

The addressing model that the MTA deals with assumes that all systems are aware of the addresses of all other systems and how to get to them. Unfortunately, this ideal is not possible in all cases, such as when a channel connects to one or more systems that are not known to the rest of the world (for example, internal machines on a private TCP/IP network). Addresses for systems on this channel may not be legal on remote systems outside of the site. If you want to be able to reply to such addresses, they must contain a source route that tells remote systems to route messages through the local machine. The local machine can then (automatically) route the messages to these machines.

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

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

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

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

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

Address Rewriting Upon Message Dequeue (connectalias, connectcanonical)

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

- The `connectalias` keyword tells the MTA to deliver to whatever host is listed in the recipient address. This is the default. The keyword `connectcanonical` forces the MTA to run the address through the rewrite rules one additional time and use the resulting host.

Channel Directionality (master, slave, bidirectional)

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

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

Channel Service Periodicity (immediate, immnonurgent, periodic, period)

If a channel is capable of master-mode operations (as specified with the `master` keyword), such operations may be initiated either by a periodic service job or on demand as delivery is needed:

- `immediate`, which is the default, specifies that jobs should run on demand for messages of appropriate urgency.
- `periodic` inhibits initiation of delivery jobs on demand for the channel it is associated with, regardless of priority.

  What *appropriate urgency* means is controlled by the keywords:
- `immnonurgent` enables immediate delivery for urgent, normal, and nonurgent messages.

The default behavior (`immediate`) enables immediate processing for all but nonurgent or lower priority messages.

Delivery by periodic service jobs is always possible unless the channel is marked with the `slave` keyword. Channels capable of master-mode operation are periodically checked for pending messages by periodic service jobs. These jobs run at fixed intervals, usually every four hours, although you can change this interval. On UNIX systems, the interval is determined in the `crontab` entry for the post job.

Not all channels need service at the same intervals. For example, a channel might see little traffic and be expensive to service. Servicing such a channel at longer intervals than that of a single period between periodic jobs can lower the cost of operation without significantly affecting the quality of service.

In another case, one particular channel may see very heavy traffic and require frequent service, while other channels need servicing much less often. In this situation it may be appropriate to service the heavily used channel more often than any other.

The `period` keyword can be used to control how often a channel is serviced. This keyword must be followed by an integer value *N*. The channel is then serviced by every *Nth* service job. The default value of the `period` keyword is `1`, which means that every periodic service job checks the channel for pending messages.

Message Size Affecting Priority (urgentblocklimit, normalblocklimit, nonurgentblocklimit)

The `urgentblocklimit`, `normalblocklimit`, and `nonurgentblocklimit` keywords may be used to downgrade the priority of messages based on their size. This priority, in turn, may affect whether a message is processed immediately, or whether it is left to wait for processing until the next periodic job runs.

The `urgentblocklimit` keyword instructs the MTA to downgrade messages larger than the specified size to `normal` priority. The `normalblocklimit` keyword instructs the MTA to downgrade messages larger than the specified size to `nonurgent` priority. The `nonurgentblocklimit` keyword instructs the MTA to downgrade messages larger than the specified size to lower than `nonurgent` priority (second class priority), meaning that the messages always wait for the next periodic job for further processing.

Channel Connection Information Caching (cacheeverything, cachesuccesses, cachefailures, nocache)

SMTP channels maintain a cache containing a history of prior connection attempts. This cache is used to avoid reconnecting multiple times to inaccessible hosts, which can waste time and delay other messages. The cache normally records both connection successes and failures. Successful connection attempts are recorded to offset subsequent failures; for example, a host that succeeded before but fails now doesn't warrant as long a delay before making another connection attempt as does one that has never been tried or one that has failed previously.

However, this caching strategy is not necessarily appropriate for all situations. For example, an SMTP channel that is used to connect to a single unpredictable host does not benefit from caching. Therefore, channel keywords are provided to adjust the MTA's cache.

The `cacheeverything` keyword enables all forms of caching and is the default. `nocache` disables all caching. The `cachefailures` enables caching of connection failures but not successes. Finally, `cachesuccesses` caches only successful connections. This last keyword is equivalent to `nocache` for channels.

Number of Addresses or Message Files to Handle per Service Job or File (addrsperjob, filesperjob, maxjobs)

When a message is enqueued to a channel, the job controller normally starts one master process per channel. If the channel is processed on a periodic basis, one master process per channel is started.

A single master process might not be sufficient to ensure prompt delivery of all messages.

The `addrsperjob` and `filesperjob` keywords can be used to create additional master processes. Each of these keywords take a single positive integer parameter which specifies how many addresses or queue entries (files) 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. The effect of these keywords is maximized; the larger number computed is the number of service jobs that are actually created.

The addrsperjob keyword computes the number of service jobs to start by dividing the total number of To: addressees in all entries by the given value. 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 maxjobs keyword places an upper limit on the total number of service jobs that can be created. This keyword must be followed by an integer value; if the computed number of service jobs is greater than this value, only maxjobs processes are actually created. If maxjobs is not specified, the default for this value is 100. Normally maxjobs is set to a value that is less than or equal to the total number of jobs that can run simultaneously in whatever service queue or queues the channel uses.

For example, if a message with four recipient addresses is queued to a channel marked addrsperjob 2 and maxjobs 5, a total of two service jobs are created. But if a message with 23 recipient addresses is queued to the same channel, only five jobs are created because of the maxjobs restriction.

NOTE These keywords affect the creation of both periodic and immediate service jobs. In the case of periodic jobs, the number of jobs created is calculated from the total number of messages in the channel queue. In the case of immediate service jobs, the calculation is based only on the message being entered into the queue at the time.

The addrsperjob keyword is generally useful only on channels that provide per-address service granularity. Currently no such channels are provided with iPlanet Messaging Server 5.0. However, the functionality is provided for third party or site-supplied channels which might be able to make use of such granularity.

Multiple Addresses (multiple, addrsperfile, single, single_sys)

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

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

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

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

The `addrsperfile` keyword is used to put a limit on the maximum number of recipients that can be associated with a single message file in a channel queue, thus limiting the number of recipients that are processed in a single operation. This keyword requires a single-integer argument specifying the maximum number of recipient addresses allowed in a message file; if this number is reached, the MTA automatically creates additional message files to accommodate them. (The default multiple keyword corresponds to imposing no limit on the number of recipients in a message file.)

Expansion of Multiple Addresses (expandlimit)

Most channels support the specification of multiple recipient addresses in the transfer of each inbound message. The specification of many recipient addresses in a single message may result in delays in message transfer processing (“online” delays). If the delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

the MTA provides a special facility to force deferred ("offline") processing if more than a given number of recipient addresses are specified for a single message. Deferral of message processing can decrease online delays enormously. Note, however, that the processing overhead is only deferred, not avoided.

This special facility is activated by using a combination of the generic reprocessing channel and the `expandlimit` keyword. The `expandlimit` keyword takes an integer argument that specifies how many addresses should be accepted in messages coming from the channel before deferring processing. The default value is infinite if the `expandlimit` keyword is not specified. A value of 0 forces deferred processing on all incoming addresses from the channel.

The `expandlimit` keyword must not be specified on the local channel or the reprocessing channel itself; the results of such a specification are unpredictable. The reprocessing channel is used to perform the deferred processing and must be added to the configuration file in order for the `expandlimit` keyword to have any effect. If your configuration was built by the MTA configuration utility, then you should already have such a channel.

Multiple Subdirectories (subdirs)

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

```
tcp_local single_sys smtp subdirs 10
```

Service Job Queue Usage and Job Deferral (pool)

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. 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 queue, the first queue listed in the Job Controller configuration file.

Deferred Delivery Dates (deferred, nodeferred)

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

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

Undeliverable Message Notification Times (notices)

The `notices` keyword controls the amount of time an undeliverable message is silently retained in a given channel queue. The MTA is capable of returning a series of warning messages to the originator and, if the message remains undeliverable, the MTA eventually returns the entire message.

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 `notices` 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).

NOTE The syntax for the notices keyword uses no punctuation. For example, the default return policy is expressed as: `notices 3 6 9 12`.

The following line specifies that if messages are enqueued to the `tcp_local` channel and deferred for later reprocessing, transient failure delivery status notifications will be generated after 1 and 2 days. If the message is still not delivered after 5 days, it will be returned to its originator.

```
tcp_local charset7 us-ascii charset8 iso-8853-1 notices 1 2 3 mail.alpha.com
```

The `defaults` channel appears immediately after the first blank line in the configuration file. It is important that a blank line appear before and after the line `defaults notices....`

Returned Messages (sendpost, nosendpost, copysendpost, errsendpost)

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

The keywords `sendpost`, `copysendpost`, `errsendpost`, and `nosendpost` control the sending of failed messages to the postmaster. The keyword `sendpost` tells the MTA to send a copy of all failed messages to the postmaster unconditionally. `copysendpost` instructs the MTA to send a copy of the failure notice to the postmaster unless the originator address on the failing message is blank, in which case, the postmaster gets copies of all failed messages except those messages that are actually themselves bounces or notifications.

The keyword errsendpost instructs the MTA to send a copy of the failure notice only to the postmaster when the notice cannot be returned to the originator. No failed messages are ever sent to the postmaster if nosendpost is specified. The default, 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.

Warning Messages (warnpost, nowarnpost, copywarnpost, errwarnpost)

In addition to returning messages, the MTA sometimes sends warnings detailing messages that it has been unable to deliver. This is generally due to timeouts based on the setting of the notices channel keyword, although in some cases channel programs may produce warning messages after failed delivery attempts. The warning messages contain a description of what's wrong and how long delivery attempts will continue. In most cases they also contain the headers and the first few lines of the message in question.

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

- warnpost–Tells the MTA to send a copy of all warning messages to the postmaster unconditionally.
- copywarnpost–Instructs the MTA to send a copy of the warning to the postmaster, unless the originator address on the undelivered message is blank.

  In this case, the postmaster gets copies of all warnings of undelivered messages except for undelivered messages that are actually themselves bounces or notifications.
- errwarnpost–Instructs the MTA to send only a copy of the warning to the postmaster when the notice cannot be returned to the originator.

No warning messages are ever sent to the postmaster if nowarnpost is specified. The default, 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.

Postmaster Returned Message Content (postheadonly, postheadbody)

When a channel program or the periodic message return job returns messages to both the postmaster and the original sender, the postmaster copy can either be the entire message or just the headers. Restricting the postmaster copy to just the headers adds an additional level of privacy to user mail. However, this by itself does not guarantee message security; postmasters and system managers are typically in a position where the contents of messages can be read using `root` system privileges, if they so choose.

The keywords `postheadonly` and `postheadbody` are used to control what gets sent to the postmaster. The keyword `postheadbody` returns both the headers and the contents of the message. It is the default.The keyword `postheadonly` causes only the headers to be sent to the postmaster.

Including Altered Addresses in Notification Messages (includefinal, suppressfinal)

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

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

Triggering New Threads in Multithreaded Channels (threaddepth)

The multithreaded SMTP client sorts outgoing messages to different destinations to different threads. 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 (hence normally all handled in one thread).

Channel Protocol Selection (smtp, nosmtp)

These options 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 keyword `nosmtp` means that the channel doesn't support SMTP; all the rest of these keywords imply SMTP support.

The selection of whether or not to use the SMTP protocol is implicit for most channels; the correct protocol is chosen by the use of the appropriate channel program or programs. Some gateway systems use the Simple Mail Transfer Protocol (SMTP) described in RFC 821 as a message envelope, while others might not use an envelope format. The result is that all envelope information is derived from the RFC 822 message header, which is present in all cases. The `smtp` keyword is used to tell the channel master programs to put a batch SMTP header on the message. The `nosmtp` keyword inhibits the generation of the batch SMTP header. The `nosmtp` is the default.

The keyword `smtp` is mandatory for all SMTP channels. The keywords `smtp_cr`, `smtp_crlf`, and `smtp_lf` can be used on SMTP channels to specify the character sequences to accept as line terminators. The keyword `smtp_crlf` means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. The keyword `smtp_lf` or `smtp` means that an LF without a preceding CR is accepted. Finally, `smtp_cr` means that a CR is accepted without a following LF. It is normal to use CRLF sequences as the SMTP line terminator, and this is what the MTA always generates; this option affects only the handling of incoming material.

SMTP EHLO Command (ehlo, checkehlo, noehlo)

RFC 1651 extends SMTP to allow for the negotiation of additional commands. This is done using the new `EHLO` command, which replaces RFC 821's `HELO` command. Extended SMTP servers respond to `EHLO` by providing a list of the extensions they support. Unextended servers return an unknown command error, and the client then sends the old `HELO` command instead.

This fallback strategy normally works well with both extended and unextended servers. Problems can arise, however, with servers that do not implement SMTP according to RFC 821. In particular, some noncompliant servers are known to drop the connection on receipt of an unknown command.

The SMTP client implements a strategy whereby it attempts to reconnect and use `HELO` when any server drops the connection on receipt of an `EHLO`. However, this strategy may not work if the remote server not only drops the connection but also goes into a problematic state upon receipt of `EHLO`.

The channel keywords `ehlo`, `noehlo`, and `checkehlo` are provided to deal with such situations. `EHLO` tells the MTA to use the `ehlo` command on all initial connection attempts. The keyword `noehlo` disables all use of the `EHLO` command. The keyword `checkehlo` tests the response banner returned by the remote SMTP server for the string "ESMTP." If this string is found, `EHLO` is used; if not, `HELO` is used. The default behavior is to use `EHLO` on all initial connection attempts, unless the banner line contains the string "fire away," in which case `HELO` is used.

NOTE There is no keyword corresponding to this default behavior, which lies between the behaviors resulting from the `ehlo` and `checkehlo` keywords.

Receiving an SMTP ETRN Command (allowetrn, blocketrn, domainetrn, silentetrn)

The `allowetrn`, `blocketrn`, `domainetrn`, and `silentetrn` keywords control the the MTA response when a sending SMTP client issues the `SMTP ETRN` command, requesting that the MTA attempt to deliver messages in the MTA queues. `allowetrn` is the default; the MTA will attempt to honor all `ETRN` commands. `silentetrn` tells the MTA to honor all `ETRN` commands, but without echoing the name of the channel that the domain matched and that the MTA will be attempting to run. `blocketrn` tells the MTA not to honor `ETRN` commands. `domainetrn` tells the MTA to honor only `ETRN` commands that specify a domain; it also causes the MTA not to echo back the name of the channel that the domain matched and that the MTA will be attempting to run.

Sending an SMTP ETRN Command (sendetrn, nosendetrn)

The extended SMTP command `ETRN` (RFC 1985) allows an SMTP client to request that a remote SMTP server start up processing of the remote side's message queues destined for sending to the original SMTP client; that is, it allows an SMTP client and SMTP server to negotiate "switching roles," where the side originally the sender becomes the receiver, and the side originally the receiver becomes the sender. In other words, `ETRN` provides a way to implement "polling" of remote SMTP systems for messages incoming to one's own system. This can be useful for systems that have only transient connections between each other, for example, over dial-up lines. When the connection is brought up and one side sends to the other, using the `ETRN` command, the SMTP client can also tell the remote side that it should now try to deliver any messages that needs to travel in the reverse direction.

The SMTP client specifies on the SMTP `ETRN` command line the name of the system to which to send messages (generally the SMTP client system's own name). If the remote SMTP server supports the `ETRN` command, it will trigger execution of a separate process to connect back to the named system and send any messages awaiting delivery for that named system.

The `sendetrn` and `nosendetrn` channel keywords control whether the MTA SMTP client sends an `ETRN` command at the beginning of an SMTP connection. The default is `nosendetrn`, meaning that the MTA will not send an `ETRN` command. The `sendetrn` keyword tells the MTA to send an `ETRN` command, if the remote SMTP server says it supports `ETRN`. The `sendetrn` keyword should be followed by the name of the system requesting that its messages receive a delivery attempt.

SMTP VRFY Commands (domainvrfy, localvrfy, novrfy)

These keywords control the MTA's use of the VRFY command in its SMTP client. Under normal circumstances there is no reason to issue a VRFY command as part of an SMTP dialogue. The SMTP MAIL TO command should perform the same function that VRFY does and return an appropriate error. However, servers exist that can accept any address in a MAIL TO (and bounce it later), whereas these same servers perform more extensive checking as part of a VRFY command.

The MTA can be configured to issue SMTP VRFY commands. The keyword domainvrfy causes a VRFY command to be issued with a full address (user@host) as its argument. The localvrfy keyword causes the MTA to issue a VRFY command with just the local part of the address (user). novrfy is the default.

Responding to SMTP VRFY commands (vrfyallow, vrfydefault, vrfyhide)

These keywords control the MTA SMTP server's response when a sending SMTP client issues an SMTP VRFY command. The vrfyallow keyword tells the MTA to issue a detailed, informative response. The vrfydefault tells the MTA to provide a detailed, informative response, unless the channel option HIDE_VERIFY=1 has been specified. The vrfyhide keyword tells the MTA to issue only a vague, ambiguous response. These keywords allow per-channel control of VRFY responses, as opposed to the HIDE_VERIFY option, which normally applies to all incoming TCP/IP channels handled through the same SMTP server.

TCP/IP Port Number (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.

TCP/IP MX Record Support (mx, nomx, defaultmx, randommx, nonrandommx)

Some TCP/IP networks support the use of MX (mail forwarding) records and some do not. Some TCP/IP channel programs can be configured not to use MX records if they are not provided by the network that the MTA system is connected to. The keyword randommx specifies that MX lookups should be done and MX record values of equal precedence should be processed in random order. The keyword nonrandommx specifies that MX lookups should be done and MX values of equal precedence should be processed in the same order in which they were received.

The mx keyword is currently equivalent to nonrandommx; it might change to be equivalent to randommx in a future release. The nomx keyword disables MX lookups. The defaultmx keyword specifies that mx should be used if the network says that MX records are supported. The keyword defaultmx is the default on channels that support MX lookups in any form.

Specifying a Last Resort Host (lastresort)

The lastresort keyword is used to specify a host 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.

Reverse DNS and IDENT Lookups on Incoming SMTP Connections (identtcp, identtcplimited, identtcpnumeric, identtcpsymbolic, identnone, identnonelimited, identnonenumeric, identnonesymbolic, forwardchecknone, forwardchecktag, forwardcheckdelete)

The identtcp keyword tells the MTA to perform a connection and lookup using 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.

The `identtcpsymbolic` keyword tells the MTA to perform a connection and lookup using 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 actual incoming IP number, as reported from a DNS reverse lookup; the IP number itself is not included in the `Received: header`.

The `identtcpnumeric` keyword tells the MTA to perform a connection and lookup using 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 actual incoming IP number --- no DNS reverse lookup on the IP number is performed.

NOTE The remote system must be running an `IDENT` server for the `IDENT` lookup caused by `identtcp` or `identtcpnumeric` to be useful.

Be aware that `IDENT` query attempts may incur a performance hit. Increasingly routers will “black hole” attempted connections to ports that they don’t recognize; if this happens on an `IDENT` query, then the MTA does not hear back until the connection times out (a TCP/IP package controlled time-out, typically on the order of a minute or two).

A lesser performance factor occurs when comparing `identtcp` or `identtcpsymbolic` to `identtcpnumeric`. The DNS reverse lookup called for with `identtcp` or `identtcpsymbolic` incurs some additional overhead to obtain the more user-friendly host name.

The `identnone` keyword disables this `IDENT` lookup, but does do IP to host name translation, and both IP number and host name will be included in the `Received:` header lines for the message. The `identnonesymbolic` keyword disables this `IDENT` lookup, but does do IP to host name translation; only the host name will be included in the `Received:` header lines for the message. The `identnonenumeric` keyword disables this `IDENT` lookup and inhibits the usual DNS reverse lookup translation of IP number to host name, and might result in a performance improvement at the cost of less user-friendly information in the `Received: header`. `identnone` is the default.

The `identtcplimited` and `identnonelimited` keywords have the same effect as `identtcp` and `identnone`, respectively, as far as `IDENT` lookups, reverse DNS lookups, and information displayed in `Received:` header lines. Where they differ is that with `identtcplimited` or `identnonelimited` the IP literal address is always used as the basis for any channel switching due to use of the `switchchannel` keyword, regardless of whether the DNS reverse lookup succeeds in determining a host name.

The `forwardchecknone`, `forwardchecktag`, and `forwardcheckdelete` channel keywords can modify the effects of doing reverse lookups, controlling whether the MTA does a forward lookup of an IP name found using a DNS reverse lookup, and if such forward lookups are requested what the MTA does if the forward lookup of the IP name does not match the original IP number of the connection. The `forwardchecknone` keyword is the default, and means that no forward lookup is done. The `forwardchecktag` keyword tells the MTA to do a forward lookup after each reverse lookup and to tag the IP name with an asterisk, *, if the number found using the forward lookup does not match that of the original connection. The `forwardcheckdelete` keyword tells the MTA to do a forward lookup after each reverse lookup and to ignore (delete) the reverse lookup returned name if the forward lookup of that name does not match the original connection IP address. Use the original IP address instead.

NOTE Having the forward lookup not match the original IP address is normal at many sites, where a more “generic” IP name is used for several different IP addresses.

These keywords are only useful on SMTP channels that run over TCP/IP.

Selecting an Alternate Channel for Incoming Mail (switchchannel, allowswitchchannel, noswitchchannel)

When an SMTP server accepts an incoming connection from a remote system, it must choose a channel with which to associate the connection. Normally this decision is based on the transfer used; for example, an incoming TCP/IP connection is automatically associated with the `tcp_local` channel.

This convention breaks down, however, when multiple outgoing channels with different characteristics are used to handle different systems over the same transfer. When this happens, incoming connections are not associated with the same channel as outgoing connections, and the result is that the corresponding channel characteristics are not associated with the remote system.

The switchchannel keyword provides a way to eliminate this difficulty. If switchchannel is specified on the server's initial channel (tcp_local), the name of the originating host is matched against the channel table; if it matches, the source channel changes accordingly. The source channel may change to any channel marked switchchannel or allowswitchchannel (the default). The keyword noswitchchannel specifies that no channel switching should be done to or from the channel.

Specification of switchchannel on anything other than a channel that a server associates with by default has no effect. At present, switchchannel only affects SMTP channels, but there are actually no other channels where switchchannel would be reasonable.

NOTE When the switchchannel is specified, the name of the originating host is obtained by a DNS reverse lookup translation of the IP address to host name. Consequently, this keyword is useful for setting up anti-spamming, but it may affect performance.

Host Name to Use When Correcting Incomplete Addresses (remotehost, noremotehost)

The MTA often receives from misconfigured or incompliant mailers and SMTP clients addresses that do not contain a domain name. The MTA attempts to make such addresses legal before allowing them to pass further. The MTA does this by appending a domain name to the address (for example, appends @siroe.com to mrochek). In the case of the SMTP server, however, the two logical choices for the domain name are:

- Local host name
- Remote host name reported by the client SMTP

Either of these two choices is likely to be correct, as both may occur operationally with some frequency. The use of the remote host's domain name is appropriate when dealing with improperly configured SMTP clients. The use of the local host's domain name is appropriate when dealing with a lightweight remote mail client such as a POP or IMAP client that uses SMTP to post messages.

The best that the MTA can do is to allow the choice to be made on a channel-by-channel basis. The remotehost channel keyword specifies that the remote host's name should be used. The noremotehost channel keyword specifies that the local host's name should be used. The keyword noremotehost is the default.

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

Legalizing Messages Without Recipient Header Lines (missingrecipientpolicy)

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

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

Table 5-4 `missingrecipientpolicy` Values

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

Note that the MISSING_RECIPIENT_POLICY option can be used to set an MTA system default for this behavior.

Eight-Bit Capability (eightbit, eightnegotiate, eightstrict, sevenbit)

Some transfers restrict the use of characters with ordinal values greater than 127 (decimal). Most notably, some SMTP servers strip the high bit and thus garble messages that use characters in this eight-bit range. 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. A channel should be marked `eightbit` if no such restriction exists.

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. The `eightstrict` keyword tells the MTA to reject any messages that contain unnegotiated eight-bit data.

Automatic Character Set Labeling (charset7, charset8, charsetesc)

The `MIME` specification provides a mechanism to label the character set used in a plain text message. Specifically, a `charset=` parameter can be specified as part of the `Content-type:` header line. Various character set names are defined in `MIME`, including US-ASCII (the default), ISO-8859-1, ISO-8859-2, and 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. Each keyword requires a single argument giving the character set name. The names are not checked for validity.

NOTE Character set conversion can be done only on character sets specified in the character set definition file `charsets.txt` found in the MTA table directory. Use the names defined in this file, if possible.

The `charset7` character set name is used if the message contains only seven-bit characters; `charset8` is used if eight-bit data is found in the message; `charsetesc` will be used if a message containing only seven bit data happens to contain the escape character. If the appropriate keyword is not specified, no character set name is inserted into the `Content-type:` header lines.

These character set specifications never override existing labels; that is, they have no effect if a message already has a character set label or is of a type other than text. It is usually appropriate to label MTA local channels as follows:

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

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

Message Line Length Restrictions (linelength)

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

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

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

Channel-Specific Use of the Reverse Database (reverse, noreverse)

The `reverse` keyword tells the MTA that addresses in messages queued to the channel should be checked against, and possibly modified, by the address reversal database or `REVERSE` mapping, if either exists. `noreverse` exempts addresses in messages queued to the channel from address reversal processing. The `reverse` keyword is the default.

Inner Header Rewriting (noinner, inner)

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

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

Restricted Mailbox Encoding (restricted, unrestricted)

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

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

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

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

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

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

Trimming Message Header Lines (headertrim, noheadertrim, headerread, noheaderread, innertrim, noinnertrim)

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

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

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

CAUTION Stripping away vital header information from messages may cause improper operation of the MTA. Be extremely careful when selecting headers to remove or limit. This facility exists because there are occasional situations where selected header lines must be removed or otherwise limited. Before trimming or removing any header line, be sure that you understand the usage of that header line and have considered the possible implications of its removal.

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

Encoding: Header Line (ignoreencoding, interpretencoding)

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

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

Generation of X-Envelope-to: Header Lines (x_env_to, nox_env_to)

The `x_env_to` and `nox_env_to` keywords control the generation or suppression of `X-Envelope-to` header lines on copies of messages queued to a specific channel. The `x_env_to` keyword enables generation of these header lines while the `nox_env_to` will remove such headers from enqueued messages. The default is `nox_env_to`.

Envelope to Address in Received: Header Lines (receivedfor, noreceivedfor, receivedfrom, noreceivedfrom)

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

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

Blank Envelope Return Addresses (returnenvelope)

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

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

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

Mapping Reply-to: Header Lines (usereplyto)

The `usereplyto` keyword controls the mapping of the `Reply-to:` header line.The default is `usereplyto 0`, which means to use the channel default behavior, which varies from channel to channel. Table 5-5 indicates the mapping specifications for the `Reply-to:` header line.

Table 5-5 Reply-to: Header Mapping Options

Value	Action
-1	Never map `Reply-to:` addresses to anything.
0	Use the channel default mapping of `Reply-to:` addresses; (varies from channel to channel). This is the default.
1	Map `Reply-to:` to `From:` if no usable `From:` address exists.
2	If there is a usable `Reply-to:` address, then map it to `From:`; otherwise, fall back to the `From:` address.

Mapping Resent- Header Lines Using a Gateway to Non-RFC 822 Environments (useresent)

The `useresent` keyword controls the use of `Resent-` header lines when using a gateway to environments that do not support RFC 822 header lines. This keyword takes a single integer-valued argument. Table 5-6 lists the values used for mapping the `Resent-` headers.

Table 5-6 Resent- Header Lines Mapping Options

Value	Action
+2	Use any `Resent-` header lines that are present to generate address information.
+1	Use only `Resent-From` header lines to generate address information; all other `Resent-` header lines are ignored.

Table 5-6 Resent- Header Lines Mapping Options *(Continued)*

Value	Action
0	Do not use `Resent-` header lines to generate address information. This is the default.

Comments in Address Header Lines (commentinc, commentomit, commentstrip, commenttotal)

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`, `commentomit`, `commentstrip`, and `commenttotal` keywords. The `commentinc` keyword tells the MTA to retain comments in header lines. It is the default. The keyword `commentomit` tells the MTA to remove any comments from addressing headers, for example, `To`, `From`, or `Cc` headers lines.

The keyword `commenttotal` tells the MTA to remove any comments from all header lines, including `Received:` header lines; this keyword is not normally useful or recommended. `commentstrip` tells the MTA to strip any nonatomic characters from all comment fields. These keywords can be applied to any channel.

Personal Names in Address Header Lines (personalinc, personalomit, personalstrip)

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`, `personalomit`, and `personalstrip` keywords. The keyword `personalinc` tells the MTA to retain personal names in the headers. It is the default. The keyword `personalomit` tells the MTA to remove all personal names.The keyword `personalstrip` tells the MTA to strip any nonatomic characters from all personal name fields. These keywords can be applied to any channel.

Two- or Four-Digit Date Conversion (datefour, datetwo)

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

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

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

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

Day of Week in Date Specifications (dayofweek, nodayofweek)

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

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

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

Automatic Splitting of Long Header Lines (maxheaderaddrs, maxheaderchars)

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

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

Header Alignment and Folding (headerlabelalign, headerlinelength)

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

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

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

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

Automatic Defragmentation of Message/Partial Messages (defragment, nodefragment)

The MIME standard provides the message / partial content type for breaking up messages into smaller parts. This is useful when messages have to traverse networks with size limits. Information is included in each part so that the message can be automatically reassembled after it arrives at its destination.

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

A defragment channel must be added to the MTA configuration file in order for the `defragment` keyword to have any effect. If your configuration was built by the MTA configuration utility, then you should already have such a channel.

Automatic Fragmentation of Large Messages (maxblocks, maxlines)

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

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

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

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

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

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

Absolute Message Size Limits (blocklimit, linelimit)

Although fragmentation can automatically break messages into smaller pieces, it is appropriate in some cases to reject messages larger than some administratively defined limit, (for example, to avoid service denial attacks). The `blocklimit` and `linelimit` keywords are used to impose absolute size limits. Each of these keywords must be followed by a single integer value.

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

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

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

Specify Maximum Length Header (maxprocchars)

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

Message Logging (logging, nologging)

The MTA provides facilities for logging each message as it is enqueued and dequeued. All log entries are made to the file `mail.log_current` in the log directory *server_root*/`msg-`*instance*/`log/imta/mail.log_current`. Logging is controlled on a per-channel basis. The `logging` keyword activates logging for a particular channel while the `nologging` keyword disables it.

Debugging Channel Master and Slave Programs (master_debug, nomaster_debug, slave_debug, noslave_debug)

Some channel programs include optional code to assist in debugging by producing additional diagnostic output. Two channel keywords are provided to enable generation of this debugging output on a per-channel basis. The keywords are `master_debug`, which enables debugging output in master programs, and `slave_debug`, which enables debugging output in slave programs. Both types of debugging output are disabled by default, corresponding to `nomaster_debug` and `noslave_debug`.

When activated, debugging output ends up in the log file associated with the channel program. The location of the log file may vary from program to program. Log files are usually kept in the MTA log directory. Master programs usually have log file names of the form `x_master.log`, where `x` is the name of the channel; `slave` programs usually have log file names of the form `x_slave.log`. Also, some channel programs, notably TCP/IP and fax channel programs, may produce additional log files with names:

- `err_x_master.log`
- `err_x_slave.log`
- `di_x_master.log`
- `di_x_xlave.log`
- `ph_x_master.log`
- `ph_x_slave.log`

In the case of the local channel, `master_debug` enables debugging output when sending from the local channel, and `slave_debug` enables debugging output as messages are delivered to the local channel, with output usually appearing in the *server_root*/`msg-`*instance*/`log/imta/l_master.log`.

Delivery of Deferred Messages (serviceall, noserviceall)

Master programs normally process only a subset of the messages queued for the channel. There may be other messages that were queued to the channel at some prior time that will not be processed. However, on some channels, particularly those that only provide a link to a single mail component, this sort of operation may be inappropriate: if the immediate delivery job is successful in connecting to the mail component it may be able to easily process all the messages that are queued.

The `serviceall` and `noserviceall` keywords control this behavior. `noserviceall`, the default, indicates that the master program should only process the messages that were queued to process after its inception. `serviceall` specifies that the master program should attempt to process all messages queued to the channel each time it runs.

It may be tempting to indulge in use of `serviceall` on most or all channels. Be warned, however, that use of `serviceall` is probably not suitable for most channels that connect to multiple remote systems, or channels that entail lots of per-message overhead. If `serviceall` is used on such channels it may cause a dramatic increase in network and message processing overhead and the net result may be slower message processing overall.

Note that these keywords do not change the order in which message processing occurs. Immediate jobs always attempt to process the messages they were created to process prior to turning to other messages that are also in the channel queue.

Sensitivity checking (sensitivitynormal, sensitivitypersonal, sensitivityprivate, sensitivitycompanyconfidential)

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

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

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

SMTP AUTH (maysaslserver, mustsaslserver, nosasl, nosaslserver, saslswitchchannel)

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

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

Use `saslswitchchannel` to cause incoming connections to be switched to a specified channel upon a client's successful use of SASL. It takes a required value, specifying the channel to which to switch.

Verify the Domain on MAIL FROM: Is In the DNS (mailfromdnsverify, nomailfromdnsverify)

Setting `mailfromdnsverify` on an incoming TCP/IP channel causes the MTA to verify that an entry in the DNS exists for the domain used on the SMTP `MAIL FROM` command, and to reject the message if no such entry exists. `nomailfromdnsverify` is the default and means that no such check is performed.

Note that performing DNS checks on the return address domain may result in rejecting some valid messages (for example, from legitimate sites that have not yet registered their domain name, or at times of bad information in the DNS); it is contrary to the spirit of being generous in what you accept and getting the e-mail through, expressed in RFC 1123, Requirements for Internet Hosts. However, some sites might want to perform such checks in cases where junk email (SPAM) is being sent with forged email addresses from non-existent domains.

Channel Operation Type (submit)

The submit keyword may be used to mark a 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.

Filter File Location (filter, nofilter, destinationfilter, nodestinationfilter, sourcefilter, nosourcefilter, fileinto, nofileinto)

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

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

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

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

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

Use authenticated address from SMTP AUTH in header (authrewrite)

The `authrewrite` channel keyword may be used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers. Normally the `SMTP AUTH` information is used, though this may be overridden via the `FROM_ACCESS` mapping.

Transport Layer Security (maytls, maytlsclient, maytlsserver, musttls, musttlsclient, musttlsserver, notlsclient, notlsserver, tlsswitchchannel)

The `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, and `tlsswitchchannel` channel keywords are used to configure TLS use during the SMTP protocol by SMTP based channels such as TCP/IP channels. `notls` is the default, and means that TLS will not be permitted or attempted. It assumes the `notlsclient` keyword, which means that TLS use will not be attempted by the MTA SMTP client on outgoing connections and the `notlsserver` keyword, which means that TLS use will not be permitted by the MTA SMTP server on incoming connections. Specifying maytls causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections. It assumes `maytlsclient`, which means that the MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS, and `maytlsserver`, which means that the MTA SMTP server will advertise support for the `STARTTLS` extension and will allow TLS use when receiving messages. Specifying `musttls` will cause the MTA to insist upon TLS in both outgoing and incoming connections; email will not be exchanged with remote systems that fail to successfully negotiate TLS use. It assumes `musttlsclient`, which means that the MTA SMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP servers that do not successfully negotiate TLS use (the MTA will issue the `STARTTLS` command and that command must succeed), and `musttlsserver`, which means that the MTA SMTP server will advertise support for the `STARTTLS` extension and will insist upon TLS use when receiving incoming messages and will not accept messages from clients that do not successfully negotiate TLS use. The `tlsswitchchannel` keyword is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. It takes a required value, specifying the channel to which to switch.

The Alias File

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

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

The format of the file is as follows:

```
user@domain: <address>

user@domain: <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 local channel option file must be stored in the MTA configuration directory and named `native_option` (for example, *server_root*`/msg-`*instance*`/imta/config/native_option`).

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

```
option=value
```

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

Table 5-7 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.
`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 local channel program should attempt. If the file can not be opened after the number of retries specified, the messages will remain in the local queue and the next run of the local channel will attempt to deliver the new messages again. The `REPEAT_COUNT` option controls how many times the channel programs will 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).

SMTP Channel Option Files

An option file may be used to control various characteristics of TCP / IP channels. Such an option file must be stored in the MTA configuration directory (*server_root*/msg-*instance*/imta/config) and named *x_option*, where *x* is the name of the channel.

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 an integer, 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 5-8.

Table 5-8 SMTP Channel Options

Option	**Description**
`ALLOW_ETRNS_PER_SESSION` (integer)	Limits the number of ETRN commands accepted per session. The default is 1.
`ALLOW_REJECTIONS_BEFORE_DEFERRAL` (integer)	Set a limit on the number of bad RCPT TO: addresses that will be allowed during a single session. That is, after the specified number of To: addresses have been rejected, all subsequent recipients, good or bad, will be rejected with a 4xx error.
`ALLOW_TRANSACTIONS_PER_SESSION` (Integer)	Limits the number of messages allowed per connection. The default is no limit.
`ALLOW_RECIPIENTS_PER_TRANSACTION` (Integer)	Limits the number of recipients allowed per message. The default is no limit.
`ATTEMPT_TRANSACTIONS_PER_SESSION` (Integer)	Limits the number of messages the MTA will attempt to transfer during any one connection session.

Table 5-8 SMTP Channel Options *(Continued)*

Option	Description
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).
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).
DATA_RECEIVE_TIME (Integer)	Specifies, in minutes, how long to wait to receive data during an SMTP dialogue. The default is 60.
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_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_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.

Table 5-8 SMTP Channel Options *(Continued)*

Option	Description
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.
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.

Table 5-8 SMTP Channel Options *(Continued)*

Option	**Description**
`LOG_CONNECTION` (integer)	The `LOG_CONNECTION` option controls whether or not connection information, e.g., 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 / fail records are logged by message enqueue and dequeue agents such as the SMTP and X.400 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.

Table 5-8 SMTP Channel Options *(Continued)*

Option	**Description**
`MAIL_TRANSMIT_TIME` (Integer)	Specifies, in minutes, how long to spend transmitting the SMTP command `MAIL FROM`. The default is 10.
`MAX_CLIENT_THREADS`	An integer number indicating the maximum number of simultaneous outbound connections that the client channel program will allow. Note that multiple processes may be used for outbound connections, depending on how you have channel-processing queues set up. This option controls the number of threads per process. The default if this option is not specified is 10.
`RCPT_TRANSMIT_TIME` (Integer)	Specifies, in minutes, how long to spend transmitting the SMTP command `RCPT TO`. 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_RECV_TIME` option). The default is 0.001666.
`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_RECV_TIME` option). The default is 0.003333.

Table 5-8 SMTP Channel Options *(Continued)*

Option	Description
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 greetings.) 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.

Conversions

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

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

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

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

Character Set Conversion and Message Reformatting Mapping

One very basic mapping table in 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 done.

On many systems there is no need to do character set conversions or message reformatting and therefore this table is not needed. Situations arise, however, where character conversions must be done.

The `CHARSET-CONVERSION` mapping can also be used to alter the format of messages. Facilities are provided to convert a number of non-MIME formats into MIME. Changes to MIME encodings and structure are also possible. These options are used when messages are being relayed to systems that only support MIME or some subset of MIME. And finally, conversion from MIME into non-MIME formats is provided in a small number of cases.

The MTA will probe the `CHARSET-CONVERSION` mapping table in two different ways. The first probe is used to determine whether or not the MTA should reformat the message and if so, what formatting options should be used. (If no reformatting is specified the MTA does not bother to check for specific character set conversions.) The input string for this first probe has the general form:

```
IN-CHAN=in-channel;OUT-CHAN=out-channel;CONVERT
```

Here *in-channel* is the name of the source channel (where the message comes from) and *out-channel* is the name of the destination channel (where the message is going). If a match occurs the resulting string should be a comma-separated list of keywords. The keywords provided are listed in Table 5-9.

Table 5-9 Character set Conversion Keywords

Keyword	Action
`Always`	Force conversion even when conversion channel is an intermediate destination.
`Appledouble`	Convert other MacMIME formats to Appledouble format.
`Applesingle`	Convert other MacMIME formats to Applesingle format.
`BASE64`	Switch MIME encodings to BASE64.

Table 5-9 Character set Conversion Keywords *(Continued)*

Keyword	Action
`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 party (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 *iPlanet Messaging Server 5.0 Administration Guide*.

Conversion File

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

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 `IMT_CONVERSION_FILE` option in the MTA tailor file. By default, this is the file *server_root*`/msg-`*instance*`/imta/conversions`.

The MTA conversion file is a text file containing entries in a format that is modeled after MIME Content-Type parameters. Each entry consists of one or more lines grouped together; each line contains one or more `name=`*value*`;` parameter clauses. Quoting rules conform to MIME conventions for Content-Type header line parameters. Every line except the last must end with a semicolon (;). A physical line in this file is limited to 252 characters. You can split a logical line into multiple physical lines using the backslash (\) continuation character. Entries are terminated either by a line that does not end in a semicolon, one or more blank lines, or both.

The rule parameters currently provided are shown in Table 5-10. Parameters not listed in the table are ignored.

Table 5-10 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 will be deleted. (If this is the only part in a message, then a single empty text part will be 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-`*m* clause. Wildcards may be used in the argument. In particular, an argument of * means to copy all the original Content-Disposition: parameters.
`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 converter.
`IN-A1-FORMAT`	Input A1-format from enclosing message/rfc822 part.

Table 5-10 Conversion Parameters *(Continued)*

Parameter	Description
`IN-A1-TYPE`	Input A1-type from enclosing message/rfc822 part.
`IN-CHAN`	Inputs channel to match for conversion (wildcards allowed). The conversion specified by this entry will only be performed if the message is coming from the specified channel.
`IN-CHANNEL`	Synonym for `IN-CHAN`.
`IN-DESCRIPTION`	Inputs MIME Content-Description.
`IN-DISPOSITION`	Inputs MIME Content-Disposition.
`IN-DPARAMETER-DEFAULT-`*n*	Inputs 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*	Inputs MIME Content-Disposition parameter name whose value is to be checked; *n* = 0, 1, 2....
`IN-DPARAMETER-VALUE-`*n*	Inputs 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*	Inputs 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*	Inputs MIME Content-Type parameter name whose value is to be checked; *n* = 0, 1, 2....
`IN-PARAMETER-VALUE-`*n*	Inputs 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`	Inputs Subject from enclosing `MESSAGE/RFC822` part.
`IN-SUBTYPE`	Inputs 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.

Table 5-10 Conversion Parameters *(Continued)*

Parameter	Description
`IN-TYPE`	Inputs 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`	
`ORIGINAL-HEADER-FILE`	0 or 1. If set to 1, the original headers or the enclosing `MESSAGE/RFC822` part are written to the file represented by the `OUTPUT_HEADERS` symbol.
`OUT-A1-FORMAT`	Output A1-format.
`OUT-A1-TYPE`	Output A1-type.
`OUT-CHAN`	Outputs channel to match for conversion (wildcards allowed). The conversion specified by this entry will be performed only if the message is destined for the specified channel.
`OUT-CHANNEL`	Synonym for `OUT-CHAN`.
`OUT-DESCRIPTION`	Outputs MIME Content-Description if it is different than the input MIME Content-Description.
`OUT-DISPOSITION`	Outputs MIME Content-Disposition if it is different than the input MIME Content-Disposition.
`OUT-DPARAMETER-NAME-`*n*	Outputs MIME Content-Disposition parameter name; *n*=0, 1, 2...
`OUT-DPARAMETER-VALUE-`*n*	Outputs MIME Content-Disposition parameter value corresponding to `OUT-DPARAMETER-NAME-`*n*.
`OUT-MODE`	Mode in which to read the converted file. This should be one of: `BLOCK`, `RECORD`, `RECORD-ATTRIBUTE`, `TEXT`.
`OUT-ENCODING`	Encoding to apply to the converted file.
`OUT-PARAMETER-NAME-`*n*	Outputs MIME Content-Type parameter name; *n* = 0, 1, 2...
`OUT-PARAMETER-VALUE-`*n*	Outputs MIME Content-Type parameter value corresponding to `OUT-PARAMETER-NAME-`*n*.
`OUT-SUBTYPE`	Outputs MIME type if it is different than the input MIME type.
`OUT-TYPE`	Outputs MIME type if it is different than the input type.

Table 5-10 Conversion Parameters *(Continued)*

Parameter	Description
OVERRIDE-HEADER-FILE	0 or 1. If set, then headers are read from the OUTPUT_HEADERS symbol, overriding the original headers in the enclosing MESSAGE/RFC822 part.
OVERRIDE-OPTION-FILE	If set, the conversion channel reads options from the OUTPUT_OPTIONS symbol.
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 converter.
PART-NUMBER	Dotted integers: *a. b. c*... The part number of the MIME body part.
RELABEL	0 or 1. This flag is ignored during conversion channel processing.
SERVICE-COMMAND	The command to execute to perform service conversion. This parameter is required; 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 5-11 shows the basic set of environment variables available for use by the conversion command.

Table 5-11 Environment Variables used by the Conversion Channel

Environment Variable	Description
`INPUT_FILE`	Name of the file containing the original body part. The converter should read this file.
`INPUT_HEADERS`	Name of the file containing the original headers for the enclosing part. The converter should read this file.
`INPUT_TYPE`	Content type of the input message part.
`INPUT_SUBTYPE`	Content subtype of the input message part.
`INPUT_DESCRIPTION`	Content description of the input message part.
`INPUT_DISPOSITION`	Content disposition of the input message part.
`MESSAGE_HEADERS`	Name of the file containing the original headers for an enclosing message. The converter should read this file.
`OUTPUT_FILE`	Name of the file where the converter should store its output. The converter should create and write this file.
`OUTPUT_HEADERS`	Name of the file where the converter should store headers for an enclosing `MESSAGE/RFC822` part. The converter should create and write this file.
`OUTPUT_OPTIONS`	Name of the file from which the converter should read options.

Table 5-12 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 5-12 Options for passing information back to the conversion channel

Option	Description
`OUTPUT_TYPE`	Content type of the output message part.
`OUTPUT_SUBTYPE`	Content subtype of the output message part.
`OUTPUT_DESCRIPTION`	Content description of the output message part.

Table 5-12 Options for passing information back to the conversion channel *(Continued)*

Option	**Description**
`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`	Content disposition of the output message part.
`OUTPUT_ENCODING`	Content transfer encoding to use on the output message part.
`OUTPUT_MODE`	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.

Additional environment variables containing Content-Type information can be created as they are needed using the `PARAMETER-SYMBOL-`*n* facility.

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.

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 *server_root*`/msg-`*instance*`/imta/config/mappings`.) The contents of the mapping file will be incorporated into the compiled configuration.

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

File Format in the Mapping File

The mapping file consists of a series of separate tables. Each table begins with its name. Names always have an alphabetic character in the first column. The table name is followed by a required blank line, and then by the entries in the table. Entries consist of zero or more indented lines. Each entry line consists of two columns separated by one or more spaces or tabs. Any spaces within an entry must be quoted. 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 (over 252 characters) may be continued by ending them with a backslash (\). The white space between the two columns and before the first column may not be omitted.

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

Including Other Files in the Mapping File

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

```
<file-spec
```

This will effectively substitute 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 (*) will match zero or more characters, and each percent sign (%) will match 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 5-13 Mapping Pattern Wildcards

Wildcard	Description
%	Match exactly one character.

Table 5-13 Mapping Pattern Wildcards *(Continued)*

*	Match zero or more characters, with maximal or “greedy” left-to-right matching

Back match	Description
$ n*	Match the nth wildcard or glob.
$_	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.

Global wildcard	Description
$A%	Match one alphabetic character, A--Z or a--z.
$A*	Match zero or more alphabetic characters, A--Z or a--z.
$B%	Match one binary digit (0 or 1).
$B*	Match zero or more binary digits (0 or 1).
$D%	Match one decimal digit 0--9.
$D*	Match zero or more decimal digits 0--9.
$H%	Match one hexadecimal digit 0--9 or A--F.
$H*	Match zero or more hexadecimal digits 0--9 or A--F.
$O%	Match one octal digit 0--7.
$O*	Match zero or more octal digits 0--7.
$S%	Match one symbol set character, for example, 0--9, A--Z, a--z, _, $.
$S*	Match zero or more symbol set characters, for example, 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.

Table 5-13 Mapping Pattern Wildcards *(Continued)*

$[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.

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

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

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

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

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

IPv4 Matching

With IPv4 matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits to ignore when checking for a match. For instance,

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

will match anything in the 123.45.67.0 subnet. Or another example is that

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

will match anything in the range 123.45.67.4--123.45.67.7.

Mapping Entry Templates

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

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

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

A dollar sign followed by a digit *n* calls for a substitution; a dollar sign followed by an alphabetic character is referred to as a "metacharacter." Metacharacters themselves will not appear in the output string produced by a template. See Table 5-14 for a list of the special substitution and standard processing metacharacters. Any other metacharacters are reserved for mapping-specific applications.

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

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

Table 5-14 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.
`$|...|`	Applies specified mapping table to supplied string.
`${...}`	General database substitution.
`$[...]`	Invokes site-supplied routine; substitute in result.
Metacharacter	**Description**

Table 5-14 Mapping Template Substitutions and Metacharacters *(Continued)*

Substitution sequence	**Substitutes**
`$C`	Continues the mapping process starting with the next table entry; uses the output string of this entry as the new input string for the mapping process.
`$E`	Ends the mapping process now; uses the output string from this entry as the final result of the mapping process.
`$L`	Continues the mapping process starting with the next table entry; use the output string of this entry as the new input string; after all entries in the table are exhausted, makes one more pass, starting with the first table entry. A subsequent match may override this condition with a `$C`, `$E`, or `$R` metacharacter.
`$R`	Continues the mapping process starting with the first entry of the mapping table; uses the output string of this entry as the new input string for the mapping process.
`$?x?`	Mapping entry succeeds x percent of the time.
`$\`	Forces subsequent text to lowercase.
`$^`	Forces subsequent text to uppercase.
`$_`	Leaves subsequent text in its original case.

Wildcard Field Substitutions ($n)

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

```
PSI$%*::*    $1@$0.psi.network.org
```

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

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

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

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

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

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

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

Entry Randomly Succeeds or Fails ($?x?)

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

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

```
PORT_ACCESS

  TCP|*|25|123.45.6.78|*           $C$?25?$Y
  TCP|*|25|123.45.6.78|*           $NTry$ again$ later
```

Sequence Number Substitutions ($#...#)

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

Permitted syntax is any one of the following:

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

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

```
$#seq-file-spec#
```

The required *seq-file-spec* argument is a full file specification for an already existing MTA sequence file, where the optional *radix* and *width* arguments specify the radix (base) in which to output the sequence value, and the number of digits to output, respectively. The default radix is 10. Radices in the range -36 to 36 are also allowed;

for instance, base 36 gives values expressed with digits 0,...,9,A,...,Z. By default, the sequence value is printed in its natural width, but if the specified width calls for a greater number of digits, then the output will be padded with 0's on the left to obtain the correct number of digits.

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

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

```
touch seq-file-spec
```

or

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

A sequence number file accessed using a mapping table must be world readable in order to operate properly. You must also have an MTA user account in order to use such sequence number files.

Mapping Table Substitutions ($|...|)

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

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

General Database Substitutions (${...})

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

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

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

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

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

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

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

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

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

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

Address-Reversal Database, REVERSE Mapping and FORWARD Mapping

Address reversal is the operation consisting of converting an address from an internal form to a public, advertised form. For example, while `uid@mailhost.alpha.com` might be a valid address within the `alpha.com` domain, it might not be an appropriate address for the outside world to see. `first.last@alpha.com` is a more likely public address.

The address reversal operation applies by default to envelop From and all header addresses. This can be changed by setting the value of the REVERSE_ENVELOPE and system options. Address reversal can be turned on or off on a per-channel basis using the reverse channel keyword.

The public address for each user is specified by the mail attribute of the user entry in the directory. The same is true for distribution lists.

The reverse database contains a mapping between any valid address and this public address. It is updated and created by `imsmta dirsync`.

The reverse database is created each time you run the `imsimta dirsync` command.

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

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

Reverse mapping can also be performed on a per-channel basis. The `src_channel|` destination and `channel|` internal addresses need to be mapped to `*|tcp_local|*@*.siroe.com` and `$|@siroe.com$Y`.

If the address matches a mapping entry, the result of the mapping is tested. The resulting string will replace the address if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. If the mapping entry specifies `$D` in addition to `$Y`, the resulting string will be run through the reversal database once more; and if a match occurs, the template from the database will replace the mapping result (and hence the address).

Table 5-15 REVERSE mapping table flags

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

As an example, suppose that the internal addresses at `siroe.com` are actually of the form `user@host.siroe.com`, but, unfortunately, the user name space is such that `user@hosta.siroe.com` and `user@hostb.siroe.com` specify the same person for all hosts at `siroe.com`. Then the following, very simple `REVERSE` mapping may be used in conjunction with the address-reversal database:

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

This mapping maps addresses of the form user@anyhost.siroe.com to user@host.siroe.com. The $D metacharacter causes the address-reversal database to be consulted. The address-reversal database should contain entries of the form:

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

The reverse and noreverse channel keywords, and the MTA options USE_REVERSE_DATABASE and REVERSE_ENVELOPE might be used to control the specifics of when and how address reversal is applied. In particular, address reversal will not be applied to addresses in messages when the destination channel is marked with the noreverse keyword. If USE_REVERSE_DATABASE is set to 0, address reversal will not be used with any channel. The REVERSE_ENVELOPE option controls whether or not address reversal is applied to envelope From addresses as well as message header addresses. See the descriptions of these options and keywords for additional information on their effects. By default, the address reversal database is used if the routability scope is set to the mail server domains.

FORWARD Address Mapping

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

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

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

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

The following example illustrates the use of a complex `REVERSE` and `FORWARD` mapping. Suppose that a system or pseudo-domain named `am.sigurd.siroe.com` associated with the `native` channel produces RFC 822 addresses of the general form:

```
"lastname, firstname"@am.sigurd.siroe.com
```

or

```
"lastname,firstname"@am.sigurd.siroe.com
```

Although these addresses are perfectly legal, they often confuse other mailers that do not fully comply with RFC 822 syntax rules—mailers that do not handle quoted addresses properly, for instance. Consequently, an address format that does not require quoting tends to operate with more mailers. One such format is:

```
firstname.lastname@am.sigurd.siroe.com
```

The goals of this example mapping are to:

- Allow any of these three address formats to be used
- Present only addresses in the original format to the `mr_gateway channel`, converting formats as necessary
- Present only addresses in the new unquoted format to all other channels, converting formats as necessary

The following mapping file tables produce the results. The REVERSE mapping shown assumes that bit 3 in the MTA option USE_REVERSE_DATABASE is set.

```
REVERSE

   *|mr_gateway|"*,$ *"@am.sigurd.siroe.com $Y"$1,$ $2"@am.sigurd.nocompany.com
   *|mr_gateway|"*,*"@am.sigurd.siroe.com   $Y"$1,$ $2"@am.sigurd.nocompany.com
   *|*|"*,$ *"@am.sigurd.siroe.com        $Y$3.$2@am.sigurd.nocompany.com
   *|*|"*,*"@am.sigurd.siroe.com          $Y$3.$2@am.sigurd.nocompany.com
   *|mr_gateway|*.*@am.sigurd.siroe.com     $Y"$2,$ $1"@am.sigurd.nocompany.com
   *|*|*.*@am.sigurd.siroe.com            $Y$2.$3@am.sigurd.nocompany.com

FORWARD

   "*,$ *"@am.sigurd.siroe.com            $Y"$0,$ $1"@am.sigurd.nocompany.com
   "*,*"@am.sigurd.siroe.com              $Y"$0,$ $1"@am.sigurd.nocompany.com
   *.*@am.sigurd.siroe.com                $Y"$1,$ $0"@am.sigurd.nocompany.com
```

Option Files

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 (*server_root*/`msg-`*instance*/`imta/config/imta_tailor`). By default, this is *server_root*/`msg-`*instance*/`imta//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 or an integer, 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*.

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 in any option file.

The available options are listed in Table 5-16.

Table 5-16 Option File Options

Options	**Description**
`ACCESS_ERRORS` (Integer 0 or 1)	The MTA provides facilities to restrict access to channels on the basis of group IDs on the SunOS operating system. If `ACCESS_ERRORS` is set to 0 (the default), when an address causes an access failure the MTA will report 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 will override this default and provide a more descriptive error.
`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_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.
`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 will be 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.

Table 5-16 Option File Options *(Continued)*

Options	Description
BOUNCE_BLOCK_LIMIT	Used to force bounces of messages over the specified size to return only the message headers, rather than the full message content.
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.
COMMENT_CHARS	Sets the `comment` characters in the MTA configuration files.
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.
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 will be produced on all channels that use the QU routines. The default of 0 disables this output.
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.
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.
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_SND_OPR	Controls the production of operator messages when a message is forced into a held state because it has too many Received: header lines.
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.

Table 5-16 Option File Options *(Continued)*

Options	**Description**
ID_DOMAIN (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.
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 will be 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 bouncing messages. The default is 20.
LOG_CONNECTION (0 or 1)	Controls whether connection information—for example, the domain name of the SMTP client sending the message—is saved in the mail.log file. A value of 1 enables connection logging. A value of 0 (the default) disables it.
LOG_DELAY_BUG	Specifies the bins for delivery delay range counters.
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_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.

Table 5-16 Option File Options *(Continued)*

Options	Description
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_PROCESS	Includes the enqueuing process ID in the MTA's log entries.
LOG_SNDOPR	Controls the production of syslog messages by the MTA message logging facility.
LOG_SIZE_BINS	Specifies the bin sizes for message size range counters.
LOG_USERNAME (0 or 1)	Controls whether the user name associated with a process that enqueues mail is saved in the mail.log file. A value of 1 enables user name logging. A value of 0 (the default) disables it.
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_HEADER_BLOCK_USE (Real Number Between 0 and 1)	Controls what fraction of the available message blocks can be used by message headers.

Table 5-16 Option File Options *(Continued)*

Options	**Description**
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 will keep entirely in memory; messages larger than this size will be written to temporary files. The default is 10. For systems with lots of memory, increasing this value may provide a performance improvement.
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 will contain 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	Specify the maximum depth to which the MTA should process MIME messages. The default is 100, which means that the MTA will process up to 100 levels of message nesting.
MAX_MIME_PARTS	Specify the maximum number of MIME parts that the MTA should process in a MIME message.
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	Legalizes messages that lack any recipient headers.
NORMAL_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to non-urgent priority. This priority, in turn, may affect whether the message is processed immediately, or whether it is left to wait for processing until the next periodic job runs.

Table 5-16 Option File Options *(Continued)*

Options	Description
NON_URGENT_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: Messages above the specified size will be downgraded to lower than nonurgent priority; they will not be processed immediately and will wait for processing until the next periodic job runs. 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.
POST_DEBUG (0 or 1)	Specifies whether debugging output is produced by the MTA's periodic delivery job. If enabled with a value of 1, this output will be 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.
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.

Table 5-16 Option File Options *(Continued)*

Options	Description
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.
REVERSE_ENVELOPE (0 or 1)	Controls whether the MTA applies the address reversal to envelope From addresses as well as header addresses. This option will have no effect if the USE_REVERSE_DATABASE option is set to 0 or if the reverse database does not exist. The default is 1, which means that the MTA will attempt to apply the database to envelope From addresses. A value of 0 will disable this use of the address reversal database.
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.
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 will occur 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.

Table 5-16 Option File Options *(Continued)*

Options	Description
URGENT_BLOCK_LIMIT (Integer)	Used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to normal priority. This priority, in turn, may affect whether the message is processed immediately or left to wait for processing until the next periodic job runs. 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 will check the database if it exists. A value of 0 will disable this use of the alias database.
USE_DOMAIN_DATABASE	Controls the use of the domain database. The default (1) means that the MTA will check the database if it exists. 0
USE_ERRORS_TO (0 or 1)	Controls whether the MTA uses the information contained in Errors-to header lines when returning messages. Setting this option to 1 directs the MTA to make use of this header line. The default (0), disable uses of this header line.
USE_FORWARD_DATABASE	Control use of the forward database.
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 5-17.
USE_WARNINGS_TO (0 or 1)	Controls whether the MTA uses the information contained in Warnings-to header lines when returning messages. Setting this option to 1 directs the MTA to make use of these header lines. The default is 0, which disables use of this header line.
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 5-17 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.

Table 5-17 USE_REVERSE_DATABASE Bit Values *(Continued)*

Bit	Value	Usage
1	2	When set, address reversal is applied before addresses have had the MTA address rewriting applied to them.
2	4	When set, address reversal will be 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` Note that bit 0 is the least significant bit. The default value for USE_REVERSE_DATABASE is 5, which means that the MTA will 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.

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. This facility is completely general and may be applied to any channel; it is controlled by the `headertrim`, `noheadertrim`, `headerread`, and `noheaderread` channel keywords.

An option file can be used in addition to the channel keywords to configure the behavior of a channel. In addition, any channel can use a header option file in order to create or remove channel-specific headers in messages processed by the channel's master program.

Header option files have a different format than other MTA option files.

Header Option File Location

For header trimming to be applied upon message *dequeue*, the MTA looks in the `config` directory (*server_root*/`msg-`*instance*/`config/imta`) 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 header trimming to be applied upon message *enqueue*, the MTA looks in the `config` directory (*server_root*`/msg-`*instance*`/config/imta`) 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 5-18.

Table 5-18 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 will appear after any existing header lines of the same type. The ADD option cannot be used in conjunction with the header line type; it will be 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 will be 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.
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 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.
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.

Table 5-18 Header options *(Continued)*

Option	Description
`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 will push header lines toward the bottom of the header while negative values will 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 *server_root*`/msg-`*instance*`/config/imta` 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 5-19.

Table 5-19 tailor File Options

Option	Description
`IMTA_ADMIN_PROPERTY`	Location of the `adminserver` properties file. The `imsimta dirsync` utility reads this file to find the domains the MTA is responsible for. The default value is `adminserver.properties`.
`IMTA_ALIAS_DATABASE`	The alias database. The default is `aliasesdb`.

Table 5-19 tailor File Options *(Continued)*

Option	Description
`IMTA_ALIAS_FILE`	The MTA aliases file. Aliases not set in the directory, for example, postmaster, are set in this file. The default is `aliases`.
`IMTA_CHARSET_DATA`	Specifies where the MTA compiled character set data is located. The default is `charset_data`.
`IMTA_CHARSET_OPTION_FILE`	File used for charset conversion options. The default is `option_charset.dat.`
`IMTA_COM`	Specifies where the MTA shell scripts are located. The default is *server_root*`/bin/msg-`*instance*`/imta/bin/`.
`IMTA_CONFIG_DATA`	Compiled configuration for the MTA. The default is *server_root*`/msg-`*instance*`/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 *server_root*`/msg-`*instance*`/imta/imta.cnf`.
`IMTA_CONVERSION_FILE`	File to set rules for the conversion channel. The default is *server_root*`/msg-`*instance*`/imta/conversions`.
`IMTA_DISPATCHER_CONFIG`	The MTA dispatcher's configuration file. The default is *server_root*`/msg-`*instance*`/imta/dispatcher.cnf`.
`IMTA_DOMAIN_DATABASE`	Database used to store additional rewrite rules. The default is *server_root*`/msg-`*instance*`/imta/db/domaindb`
`IMTA_DNSRULES`	The MTA DNS configuration library. The default is *server_root*`/msg-`*instance*`/imta/lib/imdnsrules.so.`
`IMTA_FORWARD_DATABASE`	Not used.
`IMTA_GENERAL_DATABASE`	Provided for each site's customer usage. Generally, lookups can be embedded in mappings and rewrite rules. The default is *server_root*`/msg-`*instance*`/imta/generaldb`.
`IMTA_HELP`	Location of the help files for the MTA utility. The default is *server_root*`/msg-`*instance*`/imta/lib`.
`IMTA_JBC_CONFIG_FILE`	The MTA job_controller's configuration file. The default is *server_root*`/msg-`*instance*`/imta/job_controller.cnf`.
`IMTA_JBC_SERVICE`	Specifies the host and port for the Job Controller. *Do not edit this option.*
`IMTA_LANG`	Locale of the MTA's notary messages. By default it is *server_root*`/msg-`*instance*`/imta/locale/C/LC_MESSAGES`.

Table 5-19 tailor File Options *(Continued)*

Option	Description
`IMTA_LDAP_SERVER`	Specifies the location of the LDAP directory, searched by the MTA `dirsync`, `autoreply` and other programs. The list consists of one or more `ldaphost` port pairs separated by commas. Each program reads this list and connects to the first directory that it is able to connect to. It connects to port 389, if the port is not specified. The default is just `localhostname:389`.
`IMTA_LIB`	Directory where the MTA libraries and executables are stored. The default is *server_root*`/msg-`*instance*`/imta/lib/`.
`IMTA_LIBUTIL`	The MTA utility library. By default it is *server_root*`/msg-`*instance*`/lib/libimtautil.so.1`.
`IMTA_LOG`	Location of the MTA log files. The default is *server_root*`/msg-`*instance*`/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 *server_root*`/msg-`*instance*`/imta/mappings`.
`IMTA_NAME_CONTENT_FILE`	Location of file used by the MTA for content-type conversions. The default is *server_root*`/msg-`*instance*`/imta/name_content.dat`.
`IMTA_OPTION_FILE`	Name of the MTA's option file. The default is *server_root*`/msg-`*instance*`/imta/option.dat`.
`IMTA_QUEUE`	The MTA message queue directory. The default is *server_root*`/msg-`*instance*`/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 will only be 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 will only be performed every *N* times the return job runs. By default, the return job runs once every day.
`IMTA_RETURN_SYNCH_PERIOD`	Controls queue synchronization.The default value for this option is 1. If this options is set to an integer value *N*, then the associated action will only be 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 *server_root*`/msg-`*instance*`/imta/db/reversedb`.

Table 5-19 tailor File Options *(Continued)*

Option	Description
IMTA_ROOT	Base directory for the MTA installation. The default is *server_root*/msg-*instance*/imta/.
IMTA_SCRATCH	Directory where the MTA stores its backup configuration files. During a full dirsync temporary database files are also created under this directory.The default is *server_root*/msg-*instance*/imta/tmp/.
IMTA_SYNCH_CACHE_PERIOD	Controls the queue synchronization by the post program.The default value for this option is 1. If this option is set to an integer value *N*, then the associated action will only be performed every *N* times the post job runs. By default the post job runs once every four hours.
IMTA_TABLE	The MTA configuration directory. The default is *server_root*/msg-*instance*/imta/.
IMTA_USER	Name of the postmaster. The default is inetmail. If this is changed be sure to edit the *server_root*/msg-*instance*/imta/aliases file to reflect the change to the postmaster address.
IMTA_USER_PROFILE_DATABASE	Database used for storing user's vacation, forwarding, and program delivery information. The default is *server_root*/msg-*instance*/imta/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_VERSION_LIMIT_PERIOD	Controls the frequency of purging of log files by the post job. The default value for this option is 1. If this options is set to an integer value *N*, then the associated action will only be performed every *N* times the post job runs. By default the post job runs once every four hours
IMTA_WORLD_GROUP	Can perform certain privileged operations as a member of this group. The default is mail.

Dirsync Option File

This file is used to set options for the `dirsync` program that cannot be set through the command line. This file (`dirsync.opt`) should be located in the MTA configuration directory. In this file, any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored. The format of this file is:

```
option=value
```

The *value* may be either a string or an integer, depending on the option's requirements. If any of the options in this file are changed, perform a full `dirsync` after the change. The available options are as follows:

Table 5-20 `dirsync` File Options

Option	Description
`IMTA_DL_DIR`	Directory where the distribution lists member's list files are stored. Default value is *server_root*`/msg-`*instance*`/imta/dl/`.
`IMTA_DL_HASHSIZE`	Maximum number of subdirectories under the `dl` directory. This number must be a prime number. Default value is `211`.
`IMTA_PROGRAM_CONFIG`	File where information about delivery programs are stored. The default is *server_root*`/msg-`*instance*`/imta/config/program.opt`.
`IMTA_PROGRAM_DIR`	Location of the programs used for program delivery. The default is *server_root*`/msg-`*instance*`/imta/programs/`.
`USER_SPEC_INTERNAL`	Used to create aliases and domain rewrite rules for hosted domains (`%u?%d` is the default). Where `%u` is replaced by the user part and `%d` is replaced by the domain part.
`USER_SPEC`	Used to create addresses for a channel for which no spec has been specified in the channel option file. (This does not apply to the default channels.)

Autoreply Option File

This file is used for setting options for the autoreply or vacation program. This file should be located in the MTA configuration directory. In this file, any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored The format of this file is:

```
option=value
```

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

The available options are:

Table 5-21 autoreply File Options

Option	Description
`DEBUG`	Determines whether a trace file is created for each autoreply. The default is 0 and this facility is off. A value of 1 creates an autoreply trace file for each autoreply sent in the MTA log directory. A value of 3 puts more information in the trace file.
`RESEND_TIMEOUT`	If mail arrives for a recipient with autoreply on, an autoreply is not sent if a certain period has not elapsed since the last autoreply was sent from this recipient to this specific sender. This option sets the time in hours, after which an autoreply is sent to the same sender again. The default, if this option is not set, is 168 (for example, once a week).

Job Controller

The Job Controller ensures that there is a channel job running to deliver the message each time a message is enqueued to a channel. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. If a job cannot be started because the job limit for the channel or pool (i.e., the maxjobs keyword value for the channel or the Job Controller JOB_LIMIT option for the pool, respectively) has been reached, the Job Controller waits until another job has exited, then, when the job limit is no longer exceeded, starts another job.

If a message cannot be delivered on the first attempt, the message is delayed for a period of time determined by the appropriate back-off keyword. As soon as the time specified in the back-off has elapsed, the delayed message is available for delivery, and if necessary, a channel job is started to process the message.

Internally, the Job Controller maintains a set of processing pools. Various channels may be configured to "share resources" by running within the same pool; other channels may be configured to each run in an individual pool dedicated to a particular channel. Within each pool, messages are automatically sorted into different processing queues according to the message priority; higher priority messages in a pool are processed before lower priority messages in that pool.

The Job Controller's in-memory data structure of messages currently being processed and awaiting processing typically reflects the full set of message files stored on disk in the MTA queue area. However, if a backlog of message files on disk builds up large enough to exceed the Job Controller's in-memory data structure size limit (see the `MAX_MESSAGES` option), then the Job Controller tracks in-memory only a subset of the total number of message files on disk, and works for awhile only on those messages it is tracking in-memory; once a sufficient number of messages have been delivered to free up in-memory storage space, the Job Controller will automatically refresh its in-memory store (that is, scan the MTA queue area) to update its list of messages and begin processing the additional message files that meantime have been waiting patiently on disk. Such automatic rescans of the MTA queue area are not normally apparent to sites; they are automatically performed as needed. However, sites that routinely experience extremely heavy message backlogs may wish to tune the Job Controller's behavior in this respect by using the `MAX_MESSAGES` option. By increasing the `MAX_MESSAGES` option value to allow the Job Controller to use more memory, sites can reduce the occasions when message backlogs overflow the Job Controller's in-memory cache, thereby reducing the overhead involved when the Job Controller must rescan the MTA queue directory; on the other hand, when the Job Controller does need to rescan the rebuilding of the in-memory cache will take longer (the in-memory cache being bigger). Note also that, since the Job Controller must rescan the MTA queue directory every time it is started or restarted, large message backlogs (especially if a site has increased `MAX_MESSAGES` beyond its default size), mean that starts or restarts of the Job Controller will incur more overhead than starts or restarts when no such backlog exists.

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 *server_root*`/msg-`*instance*`/imta/config/` directory.

Job Controller Configuration File Format

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.

Table 5-22 shows the available options.

Table 5-22 Job Controller Configuration File Options

Option	Description
`ANON_HOST`=0 or 1	Specifies whether or not the destination host name is useful for job scheduling. The default is 1. `ANON_HOST=0` should be specified for TCP channels.

Table 5-22 Job Controller Configuration File Options *(Continued)*

Option	Description
DEBUG=*integer*	If DEBUG is set to a value other than zero, the MTA writes debugging information to a file in the *server_root*/msg-*instance*/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 pools on every pools action. • 32—Be cautious about deleting items from pools. 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.
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.
MASTER_COMMAND=*file specification*	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 master program 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 master channel 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 5-22 Job Controller Configuration File Options *(Continued)*

Option	Description
`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 will not be 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 pool directory mimicking an `imsimta cache -sync` command.
`PURGE_ARGV=`*string*	The parameter that is passed to the job specified by `PURGE_JOB`.
`PURGE_JOB=`*file_spec*	Cleans up old log files.
`PURGE_TIME=`*time_spec*	Specifies when and how often the purge job is run. By default, this is /04:00 or every four hours from when the Job Controller started.
`RETURN_ARGV=`*string*	The parameter passed to the job specified by `RETURN_JOB`.
`RETURN_JOB=`*file_spec*	A periodic job that is started by the Job Controller.
`RETURN_TIME=`*time_spec*	Specifies when and how often the return job is run. By default, this is 00:30/24:00 or every day at 12:30 AM.
`SECRET=`*file_spec*	Shared secret used to protect requests sent to the Job Controller.
`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.
`SYNCH_TIME=`*time_spec*	The Job Controller occasionally scans the pool files on disk to check for missing files. By default, this takes place every four hours, starting four hours after the Job Controller is started. The format of the *time_spec* is HH:MM/hh:mm or /hh:mm. hh.mm is the interval between the events in hours and minutes. HH:MM is the first time in a day the even should take place. For example specifying, 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.

Table 5-22 Job Controller Configuration File Options *(Continued)*

Option	Description
`TCP_PORT=`*integer*	Specifies the TCP port on which the job controller should listen for request packets. Do not change this unless the default conflicts with another TCP application on your system. If you do change this option, change the corresponding `IMTA_JBC_SERVICE` option in the MTA tailor file, *server_root*`/msg-`*instance*`/imta/config/imta_tailor`, so that it matches. The `TCP_PORT` option applies globally and is ignored if it appears in a `[CHANNEL]` or `[POOL]` section.

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, POP3, and IMAP 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 *server_root*`/msg-`*instance*`/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.

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

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

Table 5-23 shows the available options.

Table 5-23 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, or by defining the logical or environment variable system-wide to the value FFFFFFFF. The actual meaning of each bit is described in Table 5-24.

`ENABLE_RBL=`*0 or 1*	Specifying `ENABLE_RBL=1` causes the Dispatcher to compare incoming connections to the "Black Hole" list at maps.vix.com. For instance, if the Dispatcher receives a connection from 192.168.51.32, then it will attempt to obtain the IP address for the hostname 32.51.168.192.rbl.maps.vix.com. If the query is successful, the connection will be 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 will be sent before the connection is closed:
	`5.7.1 Mail from 192.168.51.32 refused, see http://maps.vix.com/rbl/`
	If you want the MTA to log such rejections, set the 24th bit of the Dispatcher debugging `DEBUG` option, `DEBUG=16%1000000`, to cause logging of the rejections to the `dispatcher.log` file; entries will take the form:
	`access_control: host a.b.c.d found on RBL list and rejected`

`HISTORICAL_TIME=`*integer*	Controls how long the expired connections (those that have been closed) and processes (those that have exited) remain listed for statistical purposes.
`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 will 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).

Table 5-23 Dispatcher configuration file options *(Continued)*

Option	Description
`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` will result in log files with the name `tcp_smtp_server_freddy.log-*`.
`MAX_CONNS=`*integer*	Affects the Dispatcher's management of connections. This value specifies a maximum number of connections that may be active on any server process.
`MAX_HANDOFFS=`*integer*	Specifies the maximum number of concurrent asynchronous hand-offs in progress that the Dispatcher will allow 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.
`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 available before the Dispatcher shuts down. 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 fewer than `MAX_SHUTDOWN` server processes for the service. This means that processes that are eligible for shutdown can continue running until a shutdown "slot" is available.

Table 5-23 Dispatcher configuration file options *(Continued)*

Option	Description
`MIN_CONNS=`*integer*	Determines the minimum number of connections that each server 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.

`PARAMETER`	The interpretation and allowed values for the `PARAMETER` option are service specific. In the case of an 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—perhaps because your internal POP and IMAP clients have been configured to use a port other than the normal port 25, thus 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 will protect 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 5-24.

Table 5-24 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.

Table 5-24 Dispatcher Debugging Bits *(Continued)*

Bit	Hexadecimal value	Decimal value	Usage
24	x 1000000	16777216	Log PORT_ACCESS denials to the dispatcher.log file.

System Parameters on Solaris

The system's heap size (`datasize`) must be enough to accommodate the Dispatcher's thread stack usage. For each Dispatcher service compute `STACKSIZE*MAX_CONNS`, and then add up the values computed for each service. The system's heap size needs to be at least twice this number.

To display the heap size (that is, default `datasize`), use the `csh` command:

```
# limit
```

or the `ksh` command

```
# ulimit -a
```

or the utility

```
# sysdef
```

Chapter 6

Messaging Multiplexor

This chapter describes the Messaging Multiplexor configuration. This chapter contains the following sections:

- Encryption (SSL) Option
- Multiplexor Configuration

Encryption (SSL) Option

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

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

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

By default, SSL is not enabled since the SSL configuration parameters (Table 6-1) are commented out. To enable SSL, un-comment and set the following parameters:

Table 6-1 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 talk SSL to the store servers. There are no default values, but ports 993 and 995 are recommended for POP and IMAP, respectively.

Table 6-1 SSL Configuration Parameters *(Continued)*

Parameter	Description
`SSLCacheDir`	SSL session cache directory. The default is the *server-root*/`mmp-`*hostname* directory.
`SSLCertFile`	Server certificate database file location (defined when you obtained a certificate for this server). The MMP requires a server certificate to offer to clients in the handshake phase of SSL. The location specified here should be absolute, not relative to the MMP installation directory. The default is *server-root*/`mmp-`*hostname*/`cert7.db.`
`SSLCertNicknames`	Nicknames of the certificates in the SSL certificate database to offer as the server certificate. The default is `Server-Cert`.

`SSLCipherSecs`	A colon-separated list of ciphers (or the string "all") representing the cipher algorithms that this server can use to encrypt SSL sessions. The client and server agree to one of them when a session is established. The available cipher specifications are:
	
	`SSL_RSA_WITH_RC4_128_ MD5`
	`SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA`
	`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
	`SSL_RSA_FIPS_WITH_DES_CBC_SHA`
	`SSL_RSA_WITH_DES_CBC_SHA`
	`SSL_RSA_EXPORT_WITH_RC4_40_MD5`
	`SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5`
	`SSL_RSA_WITH_NULL_MD5`
	
	The default is "all".

`SSLEnable`	Whether or not to enable SSL. If set to "True" or "Yes", Multiplexor will listen on both normal and SSL ports.
	
	If SSL is enabled, all of the following variables must be set. You can specify an empty parameter with empty quotes ("").
	
	`SSLPorts`
	`SSLCertFile`
	`SSLKeyFile`
	`SSLKeyPasswdFile`
	`SSLCertNicknames`
	
	The default is `yes` (SSL is enabled).

Table 6-1 SSL Configuration Parameters *(Continued)*

Parameter	**Description**
`SSLKeyFile`	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 default is *server-root*`/mmp-`*hostname*`/key3.db.`
`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 *server-root*`/mmp-`*hostname*`/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 291). 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 default is *server-root*`/mmp-`*hostname*`/secmodule.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 6-2.

Table 6-2 Messaging Multiplexor Configuration Files

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

As an example, the `LogDir` and `LogLevel` parameters can be found in all three configuration files. In `ImapProxyAService.cfg`, they are used to specify logging parameters for IMAP-related events; similarly, these parameters in `PopProxyAService.cfg` are used to configure logging 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 291.

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

Multiplexor Configuration Parameters

You control how the MMP operates by specifying various configuration parameters in the MMP configuration files.

Table 6-3 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 6-3 for more information.

Table 6-3 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 results of pre-authentication in seconds. The default AuthCacheSize is 10,000; the default AuthCacheTTL is 900.
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/IMAP before SMTP relay authentication. AuthServiceTTL represents the amount of time in seconds that an authentication record is kept good. The default for AuthService is no; the default AuthServiceTTL is 0. The AuthService parameter should almost never be turned on globally; you should configure this by virtual domain.

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`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. 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. There is no default string.
`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 `setup` installation program.

`BGMax`
`BGPenalty`
`BGMaxBadness`
`BGDecay`
`BGLinear`
`BGExcluded`
	BadGuys configuration parameters.
	
	`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).

`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`. To maintain the integrity of your system's security, you should change the default `BindPass` to a more hard-to-guess password and also select a `BindDN` that has read-only access to the directory. The default `BindDN` is `cn=Directory Manager`, and the default `BindPass` is "secret."

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`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 LDAP server. The default is `@`, so user IDs passed to LDAP and the message store servers have the form `userid@virtual.domain`.
`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. If you are using iPlanet servers and want to use the Manage Mail Account feature, you must specify this `Capability` configuration parameter to change the MMP's capability. The default Capability string is as follows (with no line breaks): `IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN LANGUAGE XSENDER X-NETSCAPE XSERVERINFO AUTH=PLAIN`
`CertMapFile`	The name of the certmap file (for SSL client-cert-based authentications). There is no default.
`ConnLimits`	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). For this to work, passwords must be stored in LDAP in plain text format. the default is `no`.
`DefaultDomain`	The default domain; this parameter is mostly used for `HostedDomains`. If it is set, the value is appended to unqualified user IDs when there is no matching `VDMap` entry for the connection.
`HostedDomains`	Boolean, whether to support HostedDomains. Defaults to Yes.

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`LdapCacheSize` `LdapCacheTTL`	The MMP can cache results of user searches. The `LdapCacheSize` parameter defines the number of cache entries; `LdapCacheTTL` defines the results of the user searches in seconds. The default `LdapCacheSize` is 10,000; the default `LdapCacheTTL` is 900.
`LdapUrl`	Pointer to the top of the site's Users / Groups directory tree. 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. For example: `ldap://ldap1 ldap2/o=isp`. The default is `ldap://`*localhost*`/o=isp`.
`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 three configuration files: `ImapProxyAService.cfg`, `PopProxyAService.cfg`, and `AService.cfg`. The default `LogDir` is *server-root*`/mmp-`*hostname*`/log` and the default `LogLevel` is 1.
`MailHostAttrs`	Space-separated list of LDAP attributes identifying the user's mail host. Multiplexor tries to connect to each server returned by the search in the order specified by the list. The default is `mailHost`.

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
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. If this is a single-processor machine, specify 0 for optimal performance. This parameter is only found in the AService.cfg configuration file. The default is 0 (the main thread does all the work).
PreAuth	Enables Global Roaming preauthentication. With preauthentication, clients authenticate to the MMP and the MMP relays authentication information to the message store. If set to Yes, preauthentication is enabled. Note that enabling preauthentication reduces server performance. The default is no.
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. The default is NULL (only userid replayed).
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=%s.

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`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 or POP services (minus the `.cfg` extension, so the defaults are `ImapProxyAService` and `PopProxyAService`, respectively). *INSTANCENAME* can also take an optional *SECTION* parameter which allows you to specify which instance of the MMP defined in the configuration file you want to start. This makes it possible to run multiple instances of POP/IMAP on different interfaces, each with different SSL certificates or other such setting all under the same MMP. The default *SECTION* is `default`. If no *INSTANCENAME* is specified, the AService daemon passes NULL to the AService DLL when the DLL is started. The `ServiceList` parameter is only found in the `AService.cfg` configuration file. The default `ServiceList` entry is shown below (all on one line): *server-root*`/bin/msg/mmp/lib/ImapProxyAService@143|993` *server-root*`/bin/msg/mmp/lib/PopProxyAService@110`
`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.

Table 6-3 Multiplexor Configuration Parameters *(Continued)*

Variable	Description
`StoreAdmin` `StoreAdminPass`	`StoreAdmin` represents the user name of the store administrator for proxy authentication during SSL. `StoreAdminPass` is the password for this store administrator. The default `StoreAdmin` is Null; however, it is recommended that you set this to `mmpstore`. There is no default `StoreAdminPass`.
`TCPAccess`	Wrap-style filter that describes TCP access control for the MMP (globally). Defaults to NULL.
`TCPAccessAttr`	Per-user attribute that contains a wrap-style filter describing the TCP access control for the user. Defaults to `mailAccessDomain`.
`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 or 600 seconds (10 minutes) for POP. The default is 1800 seconds.
`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 `@`.
`VirtualDomainFile`	The name of the file containing your virtual domain mapping. The default file is *server-root*`/mmp-`*hostname*`/vdmap.cfg`. Uncomment this line in the configuration file to enable support for virtual domains.

Glossary

A record A type of DNS record containing a host name and its associated IP address. A records are used by messaging servers on the Internet to route email. *See also* **Domain Name System (DNS)** and **MX record**.

access control A method for controlling access to a server or to folders and files on a server.

access control rules Rules specifying user permissions for a given set of directory entries or attributes.

access control list (ACL) A set of data associated with a directory that defines the permissions that users and/or groups have for accessing it.

access domain Limits access to certain Messaging Server operations from within a specified domain. For example, an access domain can be used to limit where mail for an account can be collected.

account Information that defines a specific user or user group. This information includes the user or group name, valid email address or addresses, and how and where email is delivered.

address Information in an email message that determines where and how the message must be sent. Addresses are found both in message headers and in message envelopes. Envelope addresses determine how the message gets routed and delivered; header addresses are present merely for display purposes.

address handling The actions performed by the MTA to detect errors in addressing, to rewrite addresses if necessary, and to match addresses to recipients.

addressing protocol The addressing rules that make email possible. RFC 822 is the most widely used protocol on the Internet and the protocol supported by iPlanet Messaging Server. Other protocols include X.400 and UUCP (UNIX to UNIX Copy Protocol).

address token The address element of a rewrite rule pattern.

administration privileges The set of privileges that define a users administrative role.

administration console See **Console**.

administration server administrator User who has administrative privileges to start or stop a server even when there is no Directory Server connection. The administration server administrator has restricted server tasks (typically only Restart Server and Stop Server) for all servers in a local server group. When an administration server is installed, this administrator's entry is automatically created locally (this administrator is not a user in the user directory).

administrator A user with a defined set of administrative privileges. See also **configuration administrator**, **Directory Manager**, **administration server administrator**, **server administrator**, **message store administrator**, **top-level administrator**, **domain administrator**, **organization administrator**, **family group administrator**, **mailing list owner**.

alias An alternate name of an email address.

alias file A file used to set aliases not set in a directory, such as the postmaster alias.

Allow filter A Messaging Server access-control rule that identifies clients that are to be allowed access to one or more of the following services: POP, IMAP, or HTTP. Compare **Deny filter**.

alternate address A secondary address for an account, generally a variation on the primary address. In some cases it is convenient to have more than one address for a single account.

APOP Authenticated Post Office Protocol. Similar to the Post Office Protocol (POP), but instead of using a plaintext password for authentication, it uses an encoding of the password together with a challenge string.

AUTH An SMTP command enabling an SMTP client to specify an authentication method to the server, perform an authentication protocol exchange, and, if necessary, negotiate a security layer for subsequent protocol interactions.

authentication (1) The process of proving the identity of a client user to iPlanet Messaging Server. (2) The process of proving the identity of iPlanet Messaging Server to a client or another server.

authentication certificate A digital file sent from server to client or client to server to verify and authenticate the other party. The certificate ensures the authenticity of its holder (the client or server). Certificates are not transferable.

autoreply option file A file used for setting options for autoreply, such as vacation notices.

AutoReply utility A utility that automatically responds to messages sent to accounts with the AutoReply feature activated. Every account in iPlanet Messaging Server can be configured to automatically reply to incoming messages.

backbone The primary connectivity mechanism of a distributed system. All systems that have connectivity to an intermediate system on the backbone are connected to each other. This does not prevent you from setting up systems to bypass the backbone for reasons of cost, performance, or security.

backup The process of backing up the contents of folders from the message store to a backup device. See also **restore**.

banner A text string displayed by a service such as IMAP when a client first connects to it.

base DN A distinguished name entry in the directory from which searches will occur. Also known as a search base. For example, `ou=people`, `o=siroe.com`.

Berkeley DB A transactional database store intended for high-concurrency read-write workloads, and for applications that require transactions and recoverability. iPlanet Messaging Server uses Berkeley databases for numerous purposes.

bind DN A distinguished name used to authenticate to the Directory Server when performing an operation.

body One part of an email message. Although headers and envelopes must follow a standard format, the body of the message has a content determined by the sender—the body can contain text, graphics, or even multimedia. Structured bodies follow the MIME standard.

capability A string, provided to clients, that defines the functionality available in a given IMAP service.

CA Certificate Authority. An organization that issues digital certificates (digital identification) and makes its public key widely available to its intended audience.

Certificate Authority See **CA**.

certificate-based authentication Identification of a user from a digital certificate submitted by the client. Compare **password authentication**.

certificate database A file that contains a server's digital certificate(s). Also called a certificate file.

certificate name The name that identifies a certificate and its owner.

channel The fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. Each channel consists of one or more channel programs and an outgoing message queue for storing messages that are destined to be sent to one or more of the systems associated with the channel. See also **channel block**, **channel host table, channel program**.

channel block A single channel definition. See also channel host table.

channel host table The collective set of channel definitions.

channel program Part of a channel that performs the following functions: (1) transmits messages to remote systems and deletes messages from the queue after they are sent and (2) accepts messages from remote systems placing them in the appropriate channel queues. See also **master channel program**, **slave channel program**.

ciphertext Text that has been encrypted. Opposite of **cleartext**.

cipher An algorithm used in encryption.

client A software entity that requests services or information from a server.

CNAME record A type of DNS record that maps a domain name alias to a domain name.

cleartext Unencrypted text.

client-server model A computing model in which networked computers provide specific services to other client computers. Examples include the name-server / name-resolver paradigm of the DNS and file-server / file-client relationships such as NFS and diskless hosts.

cn LDAP alias for common name.

comment character A character that, when placed at the beginning of a line, turns the line into a nonexecutable comment.

configuration administrator Person who has administrative privileges to manage servers and configuration directory data in the entire iPlanet topology. The configuration administrator has unrestricted access to all resources in the iPlanet topology. This is the only administrator who can assign server access to other administrators. The configuration administrator initially manages administrative configuration until the administrators group and its members are in place.

configuration file A file that contains the configuration parameters for a specific component of the iPlanet Messaging system.

Configuration Directory Server A Directory Server that maintains configuration information for a server or set of servers.

configutil A command-line utility for making changes to various configuration parameters stored in the directory server or in the local configuration file, `configdb`.

congestion thresholds A disk space limit that can be set by the system administrator that prevents the database from becoming overloaded by restricting new operations when system resources are insufficient.

Console A GUI (graphical user interface) that enables you to configure, monitor, maintain, and troubleshoot many iPlanet components.

cookie Text-only strings entered into the browser's memory automatically when you visit specific web sites. Cookies are programmed by the web page author. Users can either accept or deny cookies. Accepting the cookies allows the web page to load more quickly and is not a threat to the security of your machine.

counterutil   A command-line utility for displaying all counters in a counter object.

cronjob   UNIX only. A task that is executed automatically by the cron daemon at a configured time. See **crontab file**.

crontab file   UNIX only. A list of commands, one per line, that executes automatically at a given time.

daemon   A UNIX program that runs in the background, independent of a terminal, and performs a function whenever necessary. Common examples of daemon programs are mail handlers, license servers, and print daemons. On Windows NT machines, this type of program is called a service. See also **service**.

data store   A store that contains directory information, typically for an entire directory information tree.

DC tree   Domain Component tree. A directory information tree that mirrors the DNS network syntax. An example of a distinguished name in a DC tree would be `cn=billbob,dc=bridge,dc=net,o=internet`.

defragmentation   The Multipurpose Internet Mail Extensions (MIME) feature that enables a large message that has been broken down into smaller messages or fragments to be reassembled. A Message Partial Content-Type header field that appears in each of the fragments contains information that helps reassemble the fragments into one message. See also **fragmentation**.

Delegated Administrator for Messaging. A set of interfaces (GUI and CLI) that allow domain administrators to add and modify users and groups to a hosted domain.

Delegated Administrator Console   A web browser-based software console that allows domain administrators to add and modify users and groups to a hosted domain. Also allows end users to change their password, set message forwarding rules, set vacation rules, and list distribution list subscriptions.

delegated administrator server   A daemon program that handles access control to the directory by hosted domains.

delete message   The act of marking a message for deletion. The deleted message is not removed from the message store until it is expunged or purged in a separate action by the user. See also **purge message**, **expunge message**.

deliver A command-line utility that delivers mail directly to the message store accessible by POP, IMAP, or HTTP mail clients.

delivery See **message delivery**.

delivery status notification A message giving status information about a message in route to a recipient. For example, a message indicating that delivery has been delayed because of network outages.

denial of service attack A situation where an individual intentionally or inadvertently overwhelms your mail server by flooding it with messages. Your server's throughput could be significantly impacted or the server itself could become overloaded and nonfunctional.

Deny filter A Messaging Server access-control rule that identifies clients that are to be denied access to one or more of the following services: POP, IMAP, or HTTP. Compare **Allow filter**.

dereferencing an alias Specifying, in a bind or search operation, that a directory service translate an alias distinguished name to the actual distinguished name of an entry.

directory context The point in the directory tree information at which a search begins for entries used to authenticate a user and password for message store access. See also **base DN**.

directory entry A set of directory attributes and their values identified by its distinguished name. Each entry contains an object class attribute that specifies the kind of object the entry describes and defines the set of attributes it contains.

directory information tree The tree-like hierarchical structure in which directory entries are organized. Also called a DIT. DITs can be organized along the DNS (DC trees) or Open Systems Interconnect networks (OSI trees).

directory lookup The process of searching the directory for information on a given user or resource, based on that user or resource's name or other characteristic.

Directory Manager User who has administrative privileges to the directory server database. Access control does not apply this user (think of the directory manager as the directory's superuser).

directory schema The set of rules that defines the data that can be stored in the directory.

Directory Server The iPlanet directory service based on LDAP. See also **directory service**, **Lightweight Directory Access Protocol**, **Configuration Directory Server**, **User/Groups Directory Server**.

directory service A logically centralized repository of information about people and resources within an organization. See also **Lightweight Directory Access Protocol**.

directory synchronization The process of updating—that is, synchronizing—the MTA directory cache with the current directory information stored in the directory service. See also **MTA directory cache**.

disconnected state The mail client connects to the server, makes a cache copy of selected messages, then disconnects from the server.

Dispatcher The MTA component that handles connection requests for defined TCP ports. The Dispatcher is a multithreaded connection dispatching agent that permits multiple multithreaded servers to share responsibility for a given service. When using the Dispatcher, it is possible to have several multithreaded SMTP server processes running concurrently.

distinguished name The comma-separated sequence of attributes and values that specify the unique location of an entry within the directory information tree. Often abbreviated as DN.

distribution list A list of email addresses (users) that can be sent a message by specifying one email address. Also called a group. See also **expansion**, **member**, **moderator**, and **alias**.

distribution list owner An individual who is responsible for a distribution list. An owner can add or delete distribution list members. See also **distribution list**, **expansion**, **member**, and **moderator**.

DIT See **directory information tree**.

DN See distinguished name.

dn LDAP alias for distinguished name. See also **distinguished name**.

DNS See **Domain Name System**.

DNS alias A host name that the DNS server recognizes as pointing to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, `www.siroe.domain` might be an alias that points to a real machine called `realthing.siroe.domain` where the server currently exists.

DNS database A database of domain names (host names) and their corresponding IP addresses.

DNS spoofing A form of network attack in which a DNS server has been subverted to provide false information.

domain 1) A group of computers whose host names share a common suffix, the domain name. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots), for example, `corp.mktng.siroe.com`. 2) A region of administrative control.

domain administrator User who has administrative privileges to create, modify, and delete mail users, mailing lists, and family accounts in a hosted domain by using the Delegated Administrator for Messaging GUI or CLIs. By default, this user can act as a message store administrator for all messaging servers in the topology.

domain alias A domain entry that points to another domain. By using aliases, hosted domains can have several domain names.

domain hosting The ability to host one or more domains on a shared messaging server. For example, the domains `siroe.com` and `sesta.org` might both be hosted on the `siroe.net` mail server. Users send mail to and receive mail from the hosted domain—the name of the mail server does not appear in the email address.

domain name (1) A host name used in an email address. (2) A unique name that defines an administrative organization. Domains can contain other domains. Domain names are interpreted from right to left. For example, `siroe.com` is both the domain name of the Siroe Company and a subdomain of the top-level `com` domain. The `siroe.com` domain can be further divided into subdomains such as `corp.siroe.com`, and so on. See also **host name** and **fully-qualified domain name**.

Domain Name System (DNS) A distributed name resolution software that allows computers to locate other computers on a network or the Internet by domain name. The system associates standard IP addresses with host names (such as `www.siroe.com`). Machines normally get this information from a DNS server. DNS servers provide a distributed, replicated, data query service for translating hostnames into Internet addresses. See also **A record**, **MX record**, **CNAME record**.

domain organization A sub-domain below a hosted domain in the organization tree. Domain organizations are useful for companies that wish to organize their user and group entries along departmental lines.

domain part The part of an email address to the right of the @ sign. For example, `siroe.com` is the domain part of the email address `dan@siroe.com`.

domain quota The amount of space, configured by the system administrator, allocated to a domain for email messages.

domain rewrite rules See **rewrite rules**.

domain template The part of a rewrite rule that defines how the host/domain portion of an address is rewritten. It can include either a full static host/domain address or a single field substitution string, or both.

DSN. See **Delivery Status Notification**.

dsservd A daemon that accesses the database files that hold the directory information, and communicates with directory clients using the LDAP protocol.

dssetup A Directory Server preparation tool that makes an existing Directory Server ready for use by an iPlanet Messaging Server.

dynamic group A mail group defined by an LDAP search URL. Users usually join the group by setting an LDAP attribute in their directory entry.

EHLO command An SMTP command that queries a server to find out if the server supports extended SMTP commands. Defined in RFC 1869.

encryption The process of disguising information so that it cannot be deciphered (decrypted) by anyone but the intended recipient who has the code key.

enterprise network A network that consists of collections of networks connected to each other over a geographically dispersed area. The enterprise network serves the needs of a widely distributed company and is used by the company's mission-critical applications.

envelope A container for transport information about the sender and the recipient of an email message. This information is not part of the message header. Envelopes are used by various email programs as messages are moved from place to place. Users see only the header and body of a message.

envelope field A named item of information, such as `RCPT TO`, in a message envelope.

error handler A program that handles errors. In Messaging Server, issues error messages and processes error action forms after the postmaster fills them out.

Error-Handler Action form A form sent to the postmaster account that accompanies a received message that Messaging Server cannot handle. The postmaster fills out the form to instruct the server how to process the message.

error message A message reporting an error or other situation. iPlanet Messaging Server generates messages in a number of situations, notably when it gets an email message that it can't handle. Others messages, called notification errors, are for informational purposes only.

ESP Enterprise Service Provider.

ESMTP See **Extended Simple Mail Transfer Protocol**.

ETRN An SMTP command enabling a client to request that the server start the processing of its mail queues for messages that are waiting at the server for the client machine. Defined in RFC 1985.

expander Part of an electronic mail delivery system that allows a message to be delivered to a list of addressees. Mail expanders are used to implement mailing lists. Users send messages to a single address (e.g., `hacks@somehost.edu`) and the mail expander takes care of delivery to the mailboxes in the list. Also called mail exploders. See also **EXPN**.

expansion This term applies to the MTA processing of distribution lists. The act of converting a message addressed to a distribution list into enough copies for each distribution list member.

EXPN An SMTP command for expanding a mailing list. Defined in RFC 821.

expunge message The act of marking a message for deletion and then permanently removing it from the INBOX. See also **delete message**, **purge message**.

Extended Simple Mail Transfer Protocol (ESMTP) An Internet message transport protocol. ESMTP adds optional commands to the SMTP command set for enhanced functionality, including the ability for ESMTP servers to discover which commands are implemented by the remote site.

extranet The part of a company intranet that customers and suppliers can access. See also **intranet**.

facility In a Messaging Server log-file entry, a designation of the software subsystem (such as Network or Account) that generated the log entry.

failover The automatic transfer of a computer service from one system to another to provide redundant backup.

family group administrator User who has administrative privileges to add and remove family members in a family group. This user can grant family group administrative access to other members of group.

firewall A network configuration, usually both hardware and software, that forms a barrier between networked computers within an organization and those outside the organization. A firewall is commonly used to protect information such as a network's email, discussion groups, and data files within a physical building or organization site.

folder A named collection of messages. Folders can contain other folders. Also called a mailbox. See also **personal folder**, **shared folder, INBOX**.

forwarding See **message forwarding**.

FQDN See **fully-qualified domain name**.

fragmentation The Multipurpose Internet Mail Extensions (MIME) feature that allows the breaking up of a large message into smaller messages. See also **defragmentation**.

fully-qualified domain name (FQDN) The unique name that identifies a specific Internet host. See also **domain name**.

gateway The terms gateway and application gateway refer to systems that do translation from one native format to another. Examples include X.400 to / from RFC 822 electronic mail gateways. A machine that connects two or more electronic mail systems (especially dissimilar mail systems on two different networks) and transfers messages between them. Sometimes the mapping and translation can be complex, and it generally requires a store-and-forward scheme whereby the message is received from one system completely before it is transmitted to the next system after suitable translations.

greeting form A message usually sent to users when an account is created for them. This form acts as confirmation of the new account and verification of its contents.

group Same as a distribution list. See also **dynamic group**, **static group**.

group folders These contain folders for shared and group folders. See **shared folder**.

HA See **High Availability**.

hashdir A command-line utility for determining which directory contains the message store for a particular user.

header The portion of an email message that precedes the body of the message. The header is composed of field names followed by a colon and then values. Headers contain information useful to email programs and to users trying to make sense of the message. For example, headers include delivery information, summaries of contents, tracing, and MIME information; they tell whom the message is for, who sent it, when it was sent, and what it is about. Headers must be written according to RFC 822 so that email programs can read them.

header field A named item of information, such as `From:` or `To:`, in a message header. Often referred to as a "header line".

High Availability Enables the detection of a service interruption and provides recovery mechanisms in the event of a system failure or process fault. In addition, it allows a backup system to takes over the services in the event of a primary system failure.

hop A transmission between two computers.

host The machine on which one or more servers reside.

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

host name The name of a particular machine within a domain. The host name is the IP host name, which might be either a "short-form" host name (for example, mail) or a fully qualified host name. The fully qualified host name consists of two parts: the host name and the domain name. For example, `mail.siroe.com` is the machine `mail` in the domain `siroe.com`. Host names must be unique within their domains. Your organization can have multiple machines named `mail`, as long as the machines reside in different subdomains; for example, `mail.corp.siroe.com` and `mail.field.siroe.com`. Host names always map to a specific IP address. See also **domain name**, **fully-qualified domain name**, and **IP address**.

host name hiding The practice of having domain-based email addresses that don't contain the name of a particular internal host.

HTTP See **HyperText Transfer Protocol**.

hub A host that acts as the single point of contact for the system. When two networks are separated by a firewall, for example, the firewall computer often acts as a mail hub.

HyperText Transfer Protocol A standard protocol that allows the transfer of hypertext documents over the Web. iPlanet Messaging Server provides an HTTP service to support web-based email. See **Messenger Express**.

IDENT See **Identification Protocol**.

Identification Protocol A protocol that provides a means to determine the identity of a remote process responsible for the remote end of a particular TCP connection. Defined in RFC 1413.

IMAP4 See **Internet Message Access Protocol Version 4**.

imsadmin A set of command line utilities for managing domain administrators, users, and groups.

imsasm A utility that handles the saving and recovering of user mailboxes. The `imsasm` utility invokes the `imsbackup` and `imsrestore` utilities to create and interpret a data stream.

imsbackup A command-line utility for backing up the message store.

imsimta commands A set of command line utilities for performing various maintenance, testing, and management tasks for the Message Transfer Agent (MTA).

imsrestore A command-line utility for restoring the message store.

imscripter A command-line utility that talks to an IMAP server. You can use this utility to execute a command or batch of commands on IMAP folders.

INBOX The name reserved for a user's default mailbox for mail delivery. INBOX is the only folder name that is case-insensitive. For example: INBOX, Inbox, and inbox are all valid names for a users default mailbox.

installation directory The directory into which the binary (executable) files of a server are installed. For the Messaging Server, it is a subdirectory of the server root: *serverRoot*`/bin/msg/`. Compare **instance directory**, **server root**.

instance A separately executable configuration of a server or other software entity on a given host. With a single installed set of binary files, it is possible to create multiple instances of iPlanet servers that can be run and accessed independently of each other.

instance directory The directory that contains the files that define a specific instance of a server. For the Messaging Server, it is a subdirectory of the server root: *serverRoot*`/msg-`*instanceName*`/`, where *instanceName* is the name of the server as specified at installation. Compare **installation directory**, **server root**.

Internet The name given to the worldwide network of networks that uses TCP/IP protocols.

Internet Message Access Protocol Version 4 (IMAP4) A standard protocol that allows users to be disconnected from the main messaging system and still be able to process their mail. The IMAP specification allows for administrative control for these disconnected users and for the synchronization of the users' message store once they reconnect to the messaging system.

Internet Protocol (IP) The basic network-layer protocol on which the Internet and intranets are based.

internet protocol address See **IP address**.

intranet A network of TCP/IP networks within a company or organization. Intranets enable companies to employ the same types of servers and client software used for the World Wide Web for internal applications distributed over the corporate LAN. Sensitive information on an intranet that communicates with the Internet is usually protected by a firewall. See also **firewall** and **extranet**.

invalid user An error condition that occurs during message handling. When this occurs, the message store sends a communication to the MTA, the message store deletes its copy of the message. The MTA bounces the message back to the sender and deletes its copy of the message.

IP See **Internet Protocol**.

IP address A set of numbers, separated by dots, such as `198.93.93.10`, that specifies the actual location of a machine on an intranet or the Internet. A 32-bit address assigned to hosts using TCP/IP.

iPlanet Setup The installation program for all iPlanet servers and for iPlanet Console.

ISP Internet Service Provider. A company that provides Internet services to its customers including email, electronic calendaring, access to the world wide web, and web hosting.

Job Controller The MTA component responsible for scheduling and executing tasks upon request by various other MTA components.

key database A file that contains the key pair(s) for a server's certificate(s). Also called a key file.

knowledge information Part of the directory service infrastructure information. The directory server uses knowledge information to pass requests for information to other servers.

LDAP See **Lightweight Directory Access Protocol**.

LDAP Data Interchange Format (LDIF) The format used to represent Directory Server entries in text form.

LDAP referrals An LDAP entry that consists of a symbolic link (referral) to another LDAP entry. An LDAP referral consists of an LDAP host and a distinguished name. LDAP referrals are often used to reference existing LDAP data so that this data does not have to be replicated. They are also used to maintain compatibility for programs that depend on a particular entry that may have been moved.

LDAP search string A string with replaceable parameters that defines the attributes used for directory searches. For example, an LDAP search string of "uid=%s" means that searches are based on the user ID attribute.

LDAP Server A software server that maintains an LDAP directory and services queries to the directory. The iPlanet Directory Services are implementations of an LDAP Server.

LDAP server failover A backup feature for LDAP servers. If one LDAP server fails, the system can switch over to another LDAP server.

LDAP filter A way of specifying a set of entries, based on the presence of a particular attribute or attribute value.

LDBM LDAP Data Base Manager.

LDIF See **LDAP Data Interchange Format**.

Legato Networker A third-party backup utility distributed by Legato.

level A designation of logging verbosity, meaning the relative number of types of events that are recorded in log files. At a level of Emergency, for example, very few events are logged; at a level of Informational, on the other hand, very many events are logged.

Lightweight Directory Access Protocol (LDAP) Directory service protocol designed to run over TCP/IP and across multiple platforms. A simplification of the X.500 Directory Access Protocol (DAP) that allows a single point of management for storage, retrieval, and distribution of information, including user profiles, distribution lists, and configuration data across iPlanet servers. The iPlanet Directory Server uses the LDAP protocol.

listen port The port that a server uses to communicate with clients and other servers.

local part The part of an email address that identifies the recipient. See also **domain part**.

log directory The directory in which all of a service's log files are kept.

log expiration Deletion of a log file from the log directory after it has reached its maximum permitted age.

log rotation Creation of a new log file to be the current log file. All subsequent logged events are to be written to the new current file. The log file that was the previous current file is no longer written to, but remains in the log directory.

lookup Same as a search, using the specified parameters for sorting data.

mailbox A place where messages are stored and viewed. See **folder**.

mail client The programs that help users send and receive email. This is the part of the various networks and mail programs that users have the most contact with. Mail clients create and submit messages for delivery, check for new incoming mail, and accept and organize incoming mail.

mail exchange record See **MX record**.

mailing list owner A user who has administrative privileges to add members to and delete members from the mailing list.

managed object A collection of configurable attributes, for example, a collection of attributes for the directory service.

master channel program A channel program that typically initiates a transfer to a remote system. See also **slave channel program**.

master directory server The directory server that contains the data that will be replicated.

mboxutil A command-line utility for managing mail folders. This utility lists, creates, deletes, renames, or moves mailboxes (folders). It can also be used to report quota information.

MD5 A message digest algorithm by RSA Data Security. MD5 can be used to produce a short digest of data that is unique with high probability. It is mathematically extremely hard to produce a piece of data that produces the same message digest email.

member A user or group who receives a copy of an email addressed to a distribution list. See also distribution list, expansion, moderator, and owner.

message The fundamental unit of email, a message consists of a header and a body and is often contained in an envelope while it is in transit from the sender to the recipient.

message access services The protocol servers, software drivers, and libraries that support client access to the Messaging Server message store.

message delivery The act that occurs when an MTA delivers a message to a local recipient (a mail folder or a program).

message forwarding The act that occurs when an MTA sends a message delivered to a particular account to one or more new destinations as specified by the account's attributes. Forwarding may be configurable by the user. See also **message delivery**, **message routing**.

message routing The act of transferring a message from one MTA to another when the first MTA determines that the recipient is not a local account, but might exist elsewhere. Routing is normally configurable only by a network administrator. See also **message forwarding**.

Message Handling System (MHS) A group of connected MTAs, their user agents, and message stores.

message queue The directory where messages accepted from clients and other mail servers are queued for delivery (immediate or deferred).

message quota A limit defining how much disk space a particular folder can consume.

message store The database of all locally delivered messages for a Messaging server instance. Messages can be stored on a single physical disk or stored across multiple physical disks.

message store administrator User who had administrative privileges to manage the message store for a Messaging Server installation. This user can view and monitor mailboxes, and specify access control to the store. Using proxy authorization rights, this user can run certain utilities for managing the store.

message store partition A message store or subset of a message store residing on a single physical file system partition.

message submission The client User Agent (UA) transfers a message to the mail server and requests delivery.

Message Transfer Agent (MTA) A specialized program for routing and delivering messages. MTAs work together to transfer messages and deliver them to the intended recipient. The MTA determines whether a message is delivered to the local message store or routed to another MTA for remote delivery.

Messaging Multiplexor A specialized iPlanet Messaging Server that acts as a single point of connection to multiple mail servers, facilitating the distribution of a large user base across multiple mailbox hosts.

Messaging Server administrator The administrator whose privileges include installation and administration of an iPlanet Messaging Server instance.

Messenger Express A mail client that enables users to access their mailboxes through a browser-based (HTTP) interface. Messages, folders, and other mailbox information are displayed in HTML in a browser window. See also **webmail**.

mkbackupdir A utility that creates and synchronizes the backup directory with the information in the message store. It is used in conjunction with Legato Networker.

MHS See **Message Handling System**.

MIME See **Multipurpose Internet Mail Extension**.

MMP See Messaging Multiplexor.

moderator A person who first receives all email addressed to a distribution list before (A) forwarding the message to the distribution list, (B) editing the message and then forwarding it to the distribution list, or (C) not forwarding the message to the distribution list. See also **distribution list**, **expansion**, and **member**.

MoveUser A command-line utility for moving messages in a user's mail folder from one Messaging Server to another.

MTA See **Message Transfer Agent**.

MTA configuration file The file (`imta.cnf`) that contains all channel definitions for the Messaging Server as well as the rewrite rules that determine how addresses are rewritten for routing. See also **channel** and **rewrite rule**.

MTA directory cache a snapshot of the directory service information about users and groups required by the MTA to process messages. See also **directory synchronization**.

MTA hop The act of routing a message from one MTA to another.

MUA See **user agent**.

Multiplexor See **Messaging Multiplexor**.

Multipurpose Internet Mail Extension (MIME) A protocol you can use to include multimedia in email messages by appending the multimedia file in the message.

MX record Mail Exchange Record. A type of DNS record that maps one host name to another.

name resolution The process of mapping an IP address to the corresponding name. See also **DNS**.

namespace The space from which an object name is derived and understood. Files are named within the file namespace, domain components are named within the domain namespace.

naming attribute The final attribute in a directory information tree distinguished name. See also **relative distinguished name**.

naming context A specific subtree of a directory information tree that is identified by its DN. In iPlanet Directory Server, specific types of directory information are stored in naming contexts. For example, a naming context which stores all entries for marketing employees in the Siroe Corporation at the Boston office might be called `ou=mktg`, `ou=Boston`, `o=Siroe`, `c=US`.

NDN See **nondelivery notification**.

next-hop list A list of adjacent systems a mail route uses to determine where to transfer a message. The order of the systems in the next-hop list determines the order in which the mail route transfers messages to those systems.

NIS A distributed network information service containing key information about the systems and the users on the network. The NIS database is stored on the master server and all the replica or slave servers.

NIS+ A distributed network information service containing hierarchical information about the systems and the users on the network. The NIS+ database is stored on the master server and all the replica servers.

node A domain entry in the DIT.

nondelivery notification During message transmission, if the MTA does not find a match between the address pattern and a rewrite rule, the MTA sends a nondelivery report back to the sender with the original message.

notary messages Nondelivery notifications (NDNs) and delivery status notifications (DSNs) that conform to the NOTARY specifications RFC 1892.

notification message A type of message, sent to the postmaster account by the Messaging Server, that is for informational purposes and requires no action from the postmaster. Compare **error message**.

object class A template specifying the kind of object the entry describes and the set of attributes it contains. For example, iPlanet Directory Server specifies an `emailPerson` object class which has attributes such as `commonname`, `mail` (email address), `mailHost`, and `mailQuota`.

off-line state A state in which the mail client downloads messages from a server system to a client system where they can be viewed and answered. The messages might or might not be deleted from the server.

online state A state in which messages remain on the server and are remotely responded to by the mail client.

organization administrator User who had administrative privileges to create, modify, and delete mail users and mailing lists in an organization or suborganization by using the Delegated Administrator for Messaging GUI or CLIs.

OSI tree A directory information tree that mirrors the Open Systems Interconnect network syntax. An example of a distinguished name in an OSI tree would be `cn=billt,o=bridge,c=us`.

partition See **message store partition**.

password authentication Identification of a user through user name and password. Compare certificate-based authentication.

pattern A string expression used for matching purposes, such as in Allow and Deny filters.

permanent failure An error condition that occurs during message handling. When this occurs, the message store deletes its copy of an email message. The MTA bounces the message back to the sender and deletes its copy of the message.

personal folder A folder that can be read only by the owner. See also **shared folder**.

plaintext Refers to a method for transmitting data. The definition depends on the context. For example, with SSL plaintext passwords are encrypted and are therefore not sent as cleartext. With SASL, plaintext passwords are hashed, and only a hash of the password is sent as text. See also **SSL** and **SASL**.

plaintext authentication See **password authentication**.

POP3 See **Post Office Protocol Version 3**.

port number A number that specifies an individual TCP/IP application on a host machine, providing a destination for transmitted data.

postmaster account An alias for the email group and email addresses who receive system-generated messages from the Messaging Server. The postmaster account must point to a valid mailbox or mailboxes.

Post Office Protocol Version 3 (POP3) A protocol that provides a standard delivery method and that does not require the message transfer agent to have access to the user's mail folders. Not requiring access is an advantage in a networked environment, where often the mail client and the message transfer agent are on different computers.

process A self-contained, fully functional execution environment set up by an operating system. Each instance of an application typically runs in a separate process. Compare **thread**.

protocol A formal description of messages to be exchanged and rules to be followed for two or more systems to exchange information.

provisioning The process of adding, modifying or deleting entries in the iPlanet Directory Server. These entries include users and groups and domain information.

proxy The mechanism whereby one system "fronts for" another system in responding to protocol requests. Proxy systems are used in network management to avoid having to implement full protocol stacks in simple devices, such as modems.

public key encryption A cryptographic method that uses a two-part key (code) that is made up of public and private components. To encrypt messages, the published public keys of the recipients are used. To decrypt the messages, the recipients use their unpublished private keys known only to them.

purge message The process of permanently removing messages that have been deleted and are no longer referenced in user and group folders and returning the space to the message store file system. See also **delete message**, **expunge message**.

queue See **message queue**.

RC2 A variable key-size block cipher by RSA Data Security.

RC4 A stream cipher by RSA Data Security. Faster than RC2.

readership A command-line utility for collecting readership information on shared mail folders.

reconstruct A command-line utility for reconstructing mail folders.

referral A process by which the directory server returns an information request to the client that submitted it, with information about the Directory Service Agent (DSA) that the client should contact with the request. See also **knowledge information**.

regular expression A text string that uses special characters to represent ranges or classes of characters for the purpose of pattern matching.

relaying The process of passing a message from one messaging server to another messaging server.

relative distinguished name The final attribute and its value in the attribute and value sequence of the distinguished name. See also **distinguished name**.

replica directory server The directory that will receive a copy of all or part of the data.

restore The process of restoring the contents of folders from a backup device to the message store. See also **backup**.

reverse DNS lookup The process of querying the DNS to resolve a numeric IP address into the equivalent fully qualified domain name.

rewrite rules Also known as domain rewrite rules. A tool that the MTA uses to route messages to the correct host for delivery. Rewrite rules perform the following functions: (1) extract the host/domain specification from an address of an incoming message, (2) match the host/domain specification with a rewrite rule pattern, (3) rewrite the host/domain specification based on the domain template, and (4) decide which channel queue the message should be placed in.

RFC Request For Comments. The document series, begun in 1969, describes the Internet suite of protocols and related experiments. Not all (in fact very few) RFCs describe Internet standards, but all Internet standards are published as RFCs. See `http://www.imc.org/rfcs.html`.

root entry The first entry of the directory information tree (DIT) hierarchy.

router A system responsible for determining which of several paths network traffic will follow. It uses a routing protocol to gain information about the network, and algorithms to choose the best route based on several criteria known as "routing matrix." In OSI terminology, a router is a Network Layer intermediate system. See also **gateway**.

routing See **message routing**.

safe file system A file system performs logging such that if a system crashes it is possible to rollback the data to a pre-crash state and restore all data. An example of a safe file system is Veritas File System, VxFS.

SASL See **Simple Authentication and Security Layer**.

schema Definitions—including structure and syntax—of the types of information that can be stored as entries in iPlanet Directory Server. When information that does not match the schema is stored in the directory, clients attempting to access the directory might be unable to display the proper results.

SCM See **Service Control Manager**.

search base See **base DN**.

Secure Sockets Layer (SSL) A software library establishing a secure connection between two parties (client and server).

security-module database A file that contains information describing hardware accelerators for SSL ciphers. Also called `secmod`.

sendmail A common MTA used on UNIX machines. In most applications, iPlanet Messaging Server can be used as a dropin replacement for sendmail.

server administrator Person who performs server management tasks. The server administrator provides restricted access to tasks for a particular server, depending upon task ACIs. The configuration administrator must assign user access to a server. Once a user has server access permissions, that user is a server administrator who can provide server access permissions to users.

server instance The directories, programs, and utilities representing a specific server installation.

server root The directory into which all iPlanet servers associated with a given Administration Server on a given host are installed. Typically designated *serverRoot*. Compare **installation directory**, **instance directory**.

server side rules (SSR) A set of rules for enabling server-side filtering of mail. Based on the Sieve mail filtering language.

service (1) A function provided by a server. For example, iPlanet Messaging Server provides SMTP, POP, IMAP, and HTTP services. (2) A background process on Windows NT that does not have a user interface. iPlanet servers on Windows NT platforms run as services. Equivalent to **daemon**.

Service Control Manager Windows NT administrative program for managing services.

session An instance of a client-server connection.

shared folder A folder that can be read by more than one person. Shared folders have an owner who can specify read access to the folder and who can delete messages from the shared folder. The shared folder can also have a moderator who can edit, block, or forward incoming messages. Only IMAP folders can be shared. Compare **personal folder**.

Sieve A proposed language for filtering mail.

Simple Authentication and Security Layer (SASL) A means for controlling the mechanisms by which POP, IMAP or SMTP clients identify themselves to the server. iPlanet Messaging Server support for SMTP SASL use complies with RFC 2554 (ESMTP AUTH). SASL is defined in RFC 2222.

Simple Mail Transfer Protocol (SMTP) The email protocol most commonly used by the Internet and the protocol supported by the iPlanet Messaging Server. Defined in RFC 821, with associated message format descriptions in RFC 822.

single field substitution string In a rewrite rule, part of the domain template that dynamically rewrites the specified address token of the host/domain address. See also **domain template**.

single sign-on. The ability for a user to authenticate once and gain access to multiple services (mail, directory, file services, and so on).

SIZE An SMTP extension enabling a client to declare the size of a particular message to a server. The server may indicate to the client that it is or is not willing to accept the message based on the declared message size; the server can declare the maximum message size it is willing to accept to a client. Defined in RFC 1870.

slave channel program A channel program that accepts transfers initiated by a remote system. See also **master channel program**.

smart host The mail server in a domain to which other mail servers forward messages if they do not recognize the recipients.

SMTP See **Simple Mail Transfer Protocol**.

SMTP AUTH See **AUTH**.

sn Aliased directory attribute for surname.

spoofing A form of network attack in which a client attempting to access or send a message to a server misrepresents its host name.

SSL See **Secure Sockets Layer**.

SSR See **Server Side Rules**.

static group A mail group defined statically by enumerating each group member. See also **dynamic group**.

stored A command-line utility that performs daily maintenance tasks on the message store. This utility expunges and erases messages stored on disk.

subdomain A portion of a domain. For example, in the domain name `corp.siroe.com`, `corp` is a subdomain of the domain `siroe.com`. See also **host name** and **fully-qualified domain name**.

subnet The portion of an IP address that identifies a block of host IDs.

subordinate reference The naming context that is a child of the naming context held by your directory server. See also **knowledge information**.

synchronization (1) The update of data by a master directory server to a replica directory server. (2) The update of the MTA directory cache.

TCP See **Transmission Control Protocol**.

TCP/IP See **Transmission Control Protocol/Internet Protocol**.

thread A lightweight execution instance within a process.

TLS See **Transport Layer Security**.

top-level administrator User who has administrative privileges to create, modify, and delete mail users, mailing lists, family accounts, and domains in an entire Messaging Server namespace by using the Delegated Administrator for Messaging GUI or CLIs. By default, this user can act as a message store administrator for all messaging servers in the topology.

transient failure An error condition that occurs during message handling. The remote MTA is unable to handle the message when it's delivered, but may be able to later. The local MTA returns the message to the queue and schedules it for retransmission at a later time.

Transmission Control Protocol (TCP) The basic transport protocol in the Internet protocol suite that provides reliable, connection-oriented stream service between two hosts.

Transmission Control Protocol/Internet Protocol (TCP/IP) The name given to the collection of network protocols used by the Internet protocol suite. The name refers to the two primary network protocols of the suite: TCP (Transmission Control Protocol), the transport layer protocol, and IP (Internet Protocol), the network layer protocol.

Transport Layer Security (TLS). The standardized form of SSL. See also **Secure Sockets Layer**.

transport protocols Provides the means to transfer messages between MTAs, for example SMTP and X.400.

UA See **user agent**.

UBE See **Unsolicited Bulk Email**.

uid (1) User identification. A unique string identifying a user to a system. Also referred to as a userID. (2) Aliased directory attribute for userID (login name).

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

Unsolicited Bulk Email (UBE) Unrequested and unwanted email, sent from bulk distributors, usually for commercial purposes.

upper reference Indicates the directory server that holds the naming context above your directory server's naming context in the directory information tree (DIT).

user account An account for accessing a server, maintained as an entry on a directory server.

user agent (UA) The client component, such as Netscape Communicator, that allows users to create, send, and receive mail messages.

User/Groups Directory Server A Directory Server that maintains information about users and groups in an organization.

user entry or user profile Fields that describe information about each user, required and optional, examples are: distinguished name, full name, title, telephone number, pager number, login name, password, home directory, and so on.

user folders A user's email mailboxes.

user quota The amount of space, configured by the system administrator, allocated to a user for email messages.

UUCP UNIX to UNIX Copy Program. A protocol used for communication between consenting UNIX systems.

vanity domain A domain name associated with an individual user—not with a specific server or hosted domain. A vanity domain is specified by using the `MailAlternateAddress` attribute. The vanity domain does not have an LDAP entry for the domain name. Vanity domains are useful for individuals or small organizations desiring a customized domain name, without the administration overhead of supporting their own hosted domain. Also called custom domain.

/var/mail A name often used to refer to Berkeley-style inboxes in which new mail messages are stored sequentially in a single, flat text file.

Veritas Cluster Server High availability clustering software from Veritas Software with which iPlanet Messaging Server can integrate.

virtual domain (1) An ISP hosted domain. See also **hosted domain**. (2) A domain name added by the Messaging Multiplexor to a client's user ID for LDAP searching and for logging into a mailbox server.

VRFY An SMTP command for verifying a user name. Defined in RFC 821.

webmail A generic term for browser-based email services. A browser-based client—known as a “thin” client because more processing is done on the server—accesses mail that is always stored on a server. See also **Messenger Express**.

wildcard A special character in a search string that can represent one or more other characters or ranges of characters.

workgroup Local workgroup environment, where the server performs its own routing and delivery within a local office or workgroup. Interdepartmental mail is routed to a backbone server. See also **backbone**.

X.400 A message handling system standard.

Index