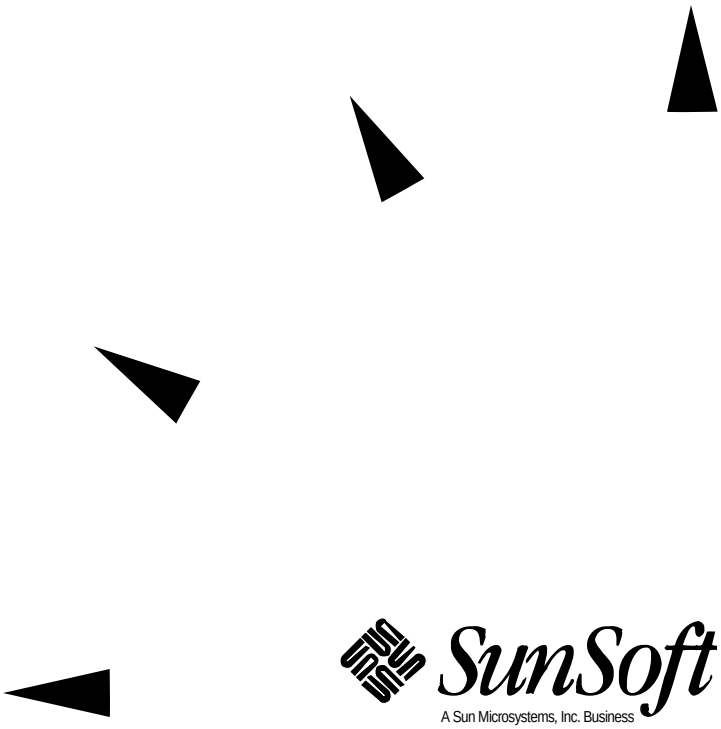
Mail Administration Guide



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

Contents

Figures

Tables

Preface

Mail Administration Guide presents the concepts and procedures required to establish and maintain electronic mail services. Special focus is given to the configuration files needed for `sendmail`.

This book assumes that you have already installed SunOS™ 5.x software and have set up the networking software that you plan to use.

Who Should Use This Book

This book is intended for the system administrator whose responsibilities include setting up and maintaining mail services. Though much of the book is directed toward the experienced system administrator, it also contains information useful to novice administrators and other readers who may be new to the Solaris™ platform.

How This Book Is Organized

Chapter 1, "Understanding Mail Services," provides an overview the mail service. The concepts and terminology necessary to set up and maintain a mail service are discussed in detail.

Chapter 2, "Setting Up and Administering Mail Services," describes the steps required to set up and administer a mail service. Troubleshooting tips are included.

Chapter 3, "Customizing sendmail Configuration Files," explains how to edit the files that `sendmail` consults.

Appendix A, "sendmail Configuration File," includes a reference copy of the generic `main.cf` file.

Appendix B, "sendmail Options," lists the command-line arguments and the configuration options to `sendmail`, and the mailer flags in table form.

Related Books

- *TCP/IP and Data Communications Administration Guide*
- *System Administration Guide, Volume I*
- *System Administration Guide, Volume II*
- *sendmail*, by Bryan Costales (O'Reilly & Associates, Inc., 1993)
- *!%@:: A Directory of Electronic Mail Addressing and Networks*, by Donnalyn Frey and Rick Adams

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	#

Understanding Mail Services 1

Setting up and maintaining an electronic mail service are complex tasks, critical to the daily operation of your network. As network administrator, you may need to expand an existing mail service or perhaps set up mail service on a new network or subnet. To help you plan a mail service for your network, this chapter provides conceptual information about mail services and briefly outlines the tasks required for setting up typical mail configurations.

Mail Services Terminology

In addition to the mail files and programs, there are many other components required to establish a mail service. The following sections define these components and some of the terminology used to describe them.

The first section defines the terminology used when discussing the software parts of the mail delivery system. The next section focuses on the functions of the hardware systems in a mail configuration.

1

Mail Services Software Terminology

Mail User Agent

The *mail user agent* is the program that acts as the interface between the user and mail transport agent, such as the `sendmail` program. The mail user agents supplied with the Solaris operating system are `/usr/bin/mail`, `/usr/bin/mailx`, `$OPENWINHOME/bin/mailtool`, and `/usr/dt/bin/dtmail`.

Mail Transport Agent

The *mail transport agent* is responsible for the routing of mail messages and resolution of mail addresses. The transport agent for the Solaris software is `sendmail`. The transport agent performs these functions:

- Accepts messages from the mail user agent
- Resolves destination addresses
- Selects a proper delivery agent to deliver the mail
- Receives incoming mail from other delivery agents

Mail Delivery Agent

A *mail delivery agent* is a program that implements a mail delivery protocol. The mail delivery agents provided with the Solaris operating environment are discussed below.

The SMTP mail delivery agent is the most commonly used protocol for TCP/IP networks. It uses the Simple Mail Transfer Protocol (SMTP) on port 25 to connect to the remote host.

The UUCP mail delivery agent uses `uux` to deliver mail.

The local mail delivery agent uses `mail.local` to deliver mail.

Mailers

A *mailer* is a sendmail specific term. A mail delivery agent can be customized. A mailer is used by sendmail to identify a specific instance of a customized mail delivery agent.

You need to specify at least one mailer in the `sendmail.cf` file of all systems in your network.

The `ether` mailer uses the SMTP protocol to transport a message. SMTP is the standard mail protocol used on the internet. This is an example of an SMTP mail header:

```
To: paul@phoenix.stateu.edu
From: Iggy.Ignatz@eng.acme.com
```

If mail is sent between two users in the same domain, the header looks like this:

```
To: Irving.Who@eng.acme.com
From: Iggy.Ignatz@eng.acme.com
```

Use SMTP for sending mail outside your domain, especially for mailboxes that you must reach through the internet.

The `smartuucp` mailer uses `uux` to deliver messages, but it formats headers with a domain-style address, and the `To:` and `CC:` lines are formatted by domain, much like the SMTP headers. The `smartuucp` headers look like this:

```
To: paul@phoenix.stateu.com
From: ignatz@eng.acme.com
```

Use `smartuucp` for UUCP mail to systems that can handle and resolve domain-style names. The sender also must be able to handle domain-style names and be able to receive replies from the Internet.

The `uucp` mailer uses an exclamation– point address in the headers. This is one of the original mailers. The headers look like this:

```
To: edu!stateu!phoenix!paul
From: acme!ignatz
```

You can define other mail delivery agents by providing a mailer specification in the `sendmail.cf` file.

Domain Names

A *domain* is a directory structure for electronic mail addressing and network address naming. The domain address has this format:

mailbox@subdomain. subdomain2.subdomain1.top-level-domain

The part of the address to the left of the `@` sign is the local address. The local address may contain information about:

- Routing using another mail transport (for example, `bob::vmsvax@gateway` or `smallberries%mill.uucp@gateway`)
- An alias (for example, `iggy.ignatz`)

The receiving mailer is responsible for determining what the local part of the address means.

The part of the address to the right of the `@` sign shows the domain address where the local address is located. A dot separates each part of the domain address. The domain can be an organization, a physical area, or a geographic region. In older forms the domain can show one or several computer systems.

Domain addresses are case-insensitive. It makes no difference whether you use uppercase, lowercase, or mixed–case letters in the domain part of an address.

The order of domain information is hierarchical—the more local the address, the closer it is to the `@` sign.

Note – British and New Zealand domains reverse this order. Most gateways automatically translate the reverse order of British and New Zealand domain names into the hierarchical order.

The larger the number of subdomains, the more detailed the information that is provided about the destination. Just as a subdirectory in a file-system hierarchy is considered to be inside the directory above, each subdomain in the mail address is considered to be inside the location to its right.

Table 1-1 shows the top-level domains in the United States.

Table 1-1 Top-level Domains in the United States

Domain	Description
Com	Commercial sites
Edu	Educational sites
Gov	Government installations
Mil	Military installations
Net	Networking organizations
Org	Nonprofit organizations

!%@:: A Directory of Electronic Mail Addressing and Networks, by Donnalyn Frey and Rick Adams (Publisher, 1993), contains a complete list of international top-level domain addresses; it is updated periodically.

For mail delivery, the network or name space domain name and the mail domain name sometimes do not match. The DNS domain name and the mail domain name must be identical. By default, the `sendmail` program strips off the first component from the network domain name to form the mail domain name. For example, if a NIS+ domain name were `bldg5.eng.acme.com`, its mail domain name would be `eng.acme.com`.

Note – Although mail domain addresses are case-insensitive, the namespace domain name is not. To reduce confusion, it is best to use lowercase characters when setting up the mail and namespace domain names.

This default rule for determining the mail domain name restricts the number of components you can have in the network domain name. Fortunately, you can define the mail domain name in the `sendmail.cf` file. You can set the `m` variable (for mail domain name) using either a `D` macro definition or an `L` macro definition. The former is a simple assignment, while the latter uses a lookup table (`sendmailvars`) maintained by the name service. The advantage of the lookup table is that you can change the mail domain name easily without having to edit the `sendmail.cf` file on each client.

Mail Address

The *mail address* contains the name of the recipient and the system to which the mail message is delivered.

When you administer a small mail system that does not use a name service, addressing mail is easy: login names uniquely identify users.

When, however, you are administering a mail system that has more than one system with mailboxes, one or more domains, or when you have a UUCP (or other) mail connection to the outside world, mail addressing becomes more complex. Mail addresses can be *route-based*, *route-independent*, or a mixture of the two.

Route-Based Addressing

Route-based addressing requires the sender of an email message to specify the local address (typically a user name) and its final destination, as well as the route that the message must take to reach its final destination. Route-based addresses are fairly common on UUCP networks, and have this format:

```
path ! host ! user
```

Whenever you see an exclamation point as part of an email address, all (or some) of the route was specified by the sender. Route-based addresses are always read from left to right.

For example, an email address that looks like this:

```
venus!acme!sierra!ignatz
```

reached user `ignatz` from the system named `venus` by going first to the address `acme`, then to `sierra`. (Note that this is an example and not an actual route.) If any of the mail handlers is out of commission, the message will be delayed or returned as undeliverable.

Route-Independent Addressing

Route-independent addressing requires the sender of an email message to specify the name of the recipient and the final destination address. Route-independent addresses usually indicate the use of a high-speed network like the Internet. In addition, newer UUCP connections frequently use domain-style names. Route-independent addresses may have this format:

```
user@host.domain
```

The increased popularity of the domain hierarchical naming scheme for computers across the country is making route-independent addresses more common. In fact, the most common route-independent address omits the host name and relies on the domain name service to properly identify the final destination of the email message:

```
user@domain
```

Route-independent addresses are read by searching for the @ sign, then reading the domain hierarchy from the right (the highest level) to the left (the most specific address to the right of the @ sign).

Mailbox

A *mailbox* is a file on a mail server that is the final destination for email messages. The name of the mailbox may be the user name or a place to put mail for someone with a specific function, like the postmaster. Mailboxes are in the `/var/mail/`*username* file, which can exist either on the user's local system or on a remote mail server. In either case, the mailbox is on the system to which the mail is delivered.

Mail should always be delivered to a local file system so that the user agent can pull mail from the mail spool and store it readily in the local mailbox. Do not use NFS-mounted file systems as the destination for a user's mailbox. In

specific, do not direct mail to a mail client that is mounting the `/var/mail` file system from a remote server. Mail for the user, in this case, should be addressed to the mail server and not to the client host name. NFS-mounted file systems can cause problems with mail delivery and handling. Clients that NFS-mount `/var/mail` go into "remote mode" and should arrange to have the server send and receive mail for them.

The `/etc/mail/aliases` file and name services like NIS and NIS+ provide mechanisms for creating aliases for electronic mail addresses, so that users do not need to know the precise local name of a user's mailbox.

Some common naming conventions for special-purpose mailboxes are shown in Table 1-2.

Table 1-2 Conventions for the Format of Mailbox Names

Format	Description
username	User names are frequently the same as mailbox names.
Firstname.*Lastname* *Firstname_Lastname* *Firstinitial*.*Lastname* *Firstinitial_Lastname*	User names may be identified as full names with a dot (or an underscore) separating the first and last names, or by a first initial with a dot (or an underscore) separating the initial and the last name.
`postmaster`	Users can address questions and report problems with the mail system to the `postmaster` mailbox. Each site and domain should have a `postmaster` mailbox.
`MAILER-DAEMON`	`sendmail` automatically routes any mail addressed to the `MAILER-DAEMON` to the postmaster.
x-`interest`	Names with dashes are likely to be a distribution list or a mailing list. This format is commonly used for net mail groups.

Table 1-2 Conventions for the Format of Mailbox Names *(Continued)*

Format	Description
x-`interest-request`	Names ending in `-request` are administrative addresses for distribution lists.
`owner-`*x*`-interest`	Names beginning with `owner-` are administrative addresses for distribution lists.
local%domain	The percent sign (%) marks a local address that is expanded when the message arrives at its destination. Most mail systems interpret mailbox names with % characters as full mail addresses. The % is replaced with an @, and the mail is redirected accordingly. Although many people use the % convention, it is not a formal standard. It is referred to as the "percent hack."

Aliases

An *alias* is an alternate name. For electronic mail, you can use aliases to assign a mailbox location or to define mailing lists.

For large sites, the mail alias is typically defines the location of a mailbox. Providing a mail alias is like providing a mail stop as part of the address for an individual at a large corporation. If you do not provide the mail stop, the mail is delivered to a central address. Extra effort is required to determine where within the building the mail is to be delivered, and the possibility of error increases. For example, if there are two people named Kevin Smith in the same building, only one of them will get mail.

Use domains and location-independent addresses as much as possible when you create mailing lists. To enhance portability and flexibility of alias files, make your alias entries in mailing lists as generic and system independent as possible. For example, if you have a user named `ignatz` on system `mars`, in domain `eng.acme.com`, create the alias `ignatz@eng` instead of `ignatz@mars`. If user `ignatz` changes the name of his system but remains within the engineering domain, you do not need to update alias files to reflect the change in system name.

When creating alias entries, type one alias per line. You should have only one entry that contains the user's system name. For example, you could create the following entries for user `ignatz`:

```
ignatz: iggy.ignatz
iggyi: iggy.ignatz
iggy.ignatz: ignatz@mars
```

You can create an alias for local names or domains. For example, an alias entry for user `fred` who has a mailbox on the system `mars` and who is in the domain `planets` could have this entry in the NIS+ aliases table:

```
fred: fred@planets
```

When creating mail lists that include users outside your domain, create the alias with the user name and the domain name. For example, if you have a user named `smallberries` on system `privet`, in domain `mgmt.acme.com`, create the alias as `smallberries@mgmt.acme.com`.

You can configure the `sendmail.cf` file to translate the email address to a fully qualified domain name when mail goes outside the user's domain.

Uses for Aliases Files

You create mail aliases for global use in the NIS+ mail_aliases table, the NIS `aliases` map, or in local `/etc/mail/aliases` files. You can also create and administer mailing lists using the same alias files.

Depending on the configuration of your mail services, you may administer aliases by using the NIS or NIS+ name service to maintain a global `aliases` database or by updating all of the local `/etc/mail/aliases` files to keep them synchronized.

Users can also create and use aliases. They can create aliases either in their local `~/.mailrc` file, which only they can use, or in their local `/etc/mail/aliases` file, which can be used by anyone. Users cannot normally create or administer NIS or NIS+ alias files.

Hardware Components of a Mail Configuration

A mail configuration requires three elements, which can be combined on the same system or provided by separate systems:

- A mail host
- At least one mail server
- Mail clients

When you want users to communicate with networks outside your domain, you must also add a fourth element, a mail gateway.

Figure 1-1 on page 11 shows a typical electronic mail configuration, using the three basic mail elements plus a mail gateway. Each element is identified and described in the following sections.

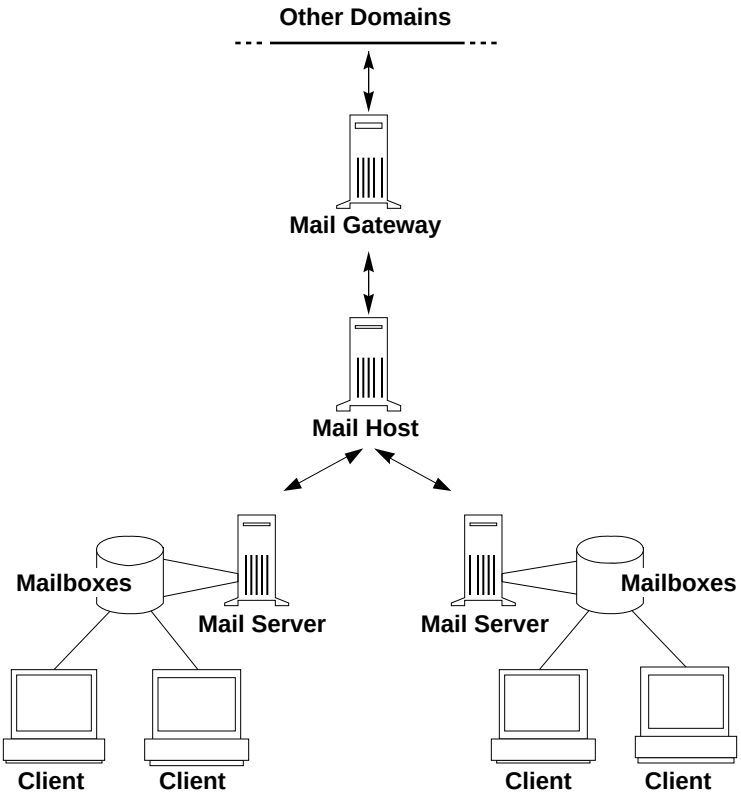


Figure 1-1 Typical Electronic Mail Configuration

Mail Host

A *mail host* is the machine that you designate as the main mail machine on your network. It is the machine to which other systems at the site forward mail that they cannot deliver. You designate a system as a mail host in the `hosts` database, by adding the word `mailhost` to the right of the IP address in the local `/etc/inet/hosts` file or in the hosts file in the name service. You must also use the `main.cf` file as the mail–configuration file on the mail host system.

A good candidate for mail host is a system on the local–area network that also has a modem for setting up PPP or UUCP links over telephone lines. Another good candidate is a system configured as a router from your network to the Internet global network. (See *TCP/IP and Data Communications Administration Guide* for more information on PPP, UUCP, and routers.) If none of the systems on your local network has a modem, designate one as the mail host.

Some sites use standalone machines that are not networked in a time-sharing configuration; that is, the standalone machine serves terminals attached to its serial ports. You can set up electronic mail for this configuration by treating the standalone system as the mail host of a one-system network.

Mail Server

A *mailbox* is a single file that contains email for a particular user. Mail is delivered to the system where the user's mailbox resides: the local machine or a remote server. A *mail server* is any system that maintains user mailboxes in its `/var/mail` directory.

The mail server routes all mail from a client. When a client sends mail, the mail server puts it in a queue for delivery. Once the mail is in the queue, a user can reboot or power down the client without losing those mail messages. When the recipient gets mail from a client, the path in the "From" line of the message contains the name of the mail server. If the recipient responds, the response goes to the user's mailbox.

If the mail server is not the user's local system, users in configurations using NFS software can mount the `/var/mail` directory by using the `/etc/vfstab` file (if they have root access) or by using the automounter. If NFS support is not available, the users can log in to the server to read their mail.

Good candidates for mail servers are systems that provide a home directory for users or that are backed up regularly.

If users on your network send other types of mail, such as PostScript™ files, audio files, or files from desktop publishing systems, you will need to allocate more space on the mail server for mailboxes.

One advantage to establishing a mail server for all mailboxes is that it makes backups very easy. Having mail spread over many systems makes it very hard to do backups. The disadvantage to storing many mailboxes on one server is that the server can be a single point of failure for many users, but the advantages of providing good backups usually make the risk worthwhile.

Mail Client

A *mail client* is any system that receives mail on a mail server and does not have a local `/var/mail` directory. This is know as the remote mode. The remote mode is enabled by adding the OR option in the `sendmail.cf` file.

You must make sure the mail client has the appropriate entry in the `/etc/vfstab` file and a mount point to mount the mailbox from the mail server. Also make sure that the alias for the client is directed to the mail server's hostname not to the client's.

Mail Gateway

The mail *gateway* is a machine that handles connections between networks running different communications protocols or communications between different networks using the same protocol. For example, a mail gateway might connect a TCP/IP network to a network running the Systems Network Architecture (SNA) protocol suite.

The simplest mail gateway to set up is one that connects two networks that use the same protocol or mailer. This system handles mail with an address for which `sendmail` can not find a recipient in your domain. If a mail gateway exists, `sendmail` uses it for sending and receiving mail outside your domain.

You can set up a mail gateway between two networks using unmatched mailers. To support this, you must customize the `sendmail.cf` file on the mail gateway system, which can be a difficult and time-consuming process.

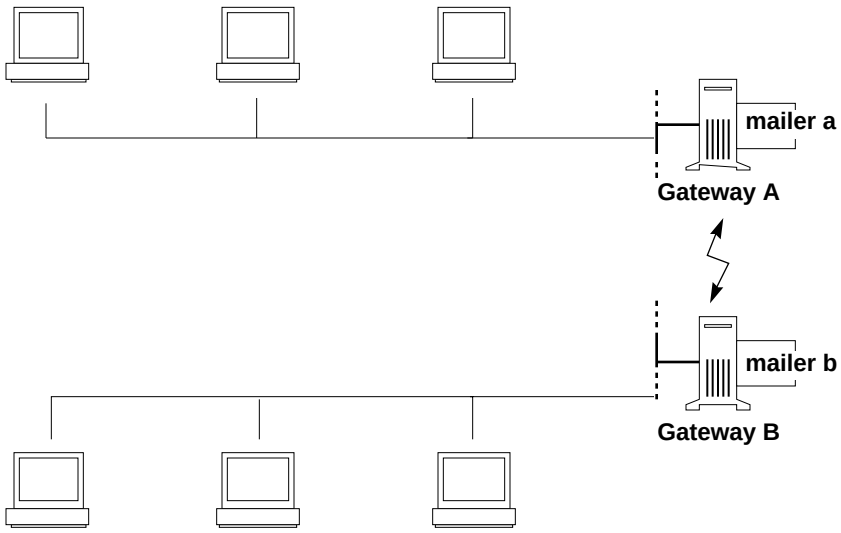


Figure 1-2 Gateway Between Different Communications Protocols

If you have to set up a mail gateway, you should find a gateway–configuration file that is close to what you need and modify it to fit your situation.

If you have a machine providing connections to the Internet, you can configure that machine as the mail gateway. Carefully consider your site's security needs before you configure a mail gateway. You may need to create a firewall gateway between your corporate network and the outside world, and set that up as the mail gateway.

Mail Service Programs and Files

Mail services include many programs and daemons that interact with each other. This section introduces the programs and the terms and concepts related to administering electronic mail. Table 1-3 shows the contents of the `/usr/bin` directory that are used for mail services.

Table 1-3 Contents of the `/usr/bin` Directory Used for Mail Services

Name	Type	Description
`mail`	File	A user agent
`mailcompat`	File	A filter to store mail in SunOS 4.1 mailbox format
`mailq`	Link	Link to `/usr/lib/sendmail`; used to list the mail queue
`mailstats`	File	A program used to read mail statistics stored in the `/etc/mail/sendmail.st` file (if present)
`mailx`	File	A user agent
`aliasadm`	File	A program to manipulate the NIS+ aliases map
`mconnect`	File	A program that connects to the mailer for address verification and debugging
`newaliases`	Link	Link to `/usr/lib/sendmail`; used to create the binary form of the aliases file
`rmail`	Link	Link to `/usr/bin/mail`; command often used to permit only the sending of mail.
`vacation`	File	A command to set up an automatic reply to mail

Table 1-4 shows the contents of the `/etc/mail` directory.

Table 1-4 Contents of the `/etc/mail` Directory

Name	Type	Description
`Mail.rc`	File	Default settings for the `mailtool` user agent
`aliases`	File	Mail–forwarding information
`aliases.dir`	File	Binary form of mail–forwarding information (created by running `newaliases`)
`aliases.pag`	File	Binary form of mail–forwarding information (created by running `newaliases`)

Table 1-4 Contents of the `/etc/mail` Directory *(Continued)*

Name	Type	Description
`mailx.rc`	File	Default settings for the `mailx` user agent
`main.cf`	File	Sample configuration file for main systems
`sendmail.cf`	File	Configuration file for mail routing
`sendmail.hf`	File	Help file used by the `SMTP HELP` command
`sendmail.pid`	File	File that lists the PID of the listening daemon
`sendmail.st`	File	The `sendmail` statistics file; if this file is present, `sendmail` logs the amount of traffic through each mailer
`subsidiary.cf`	File	Sample configuration file for subsidiary systems

Table 1-5 shows the contents of the `/usr/lib` directory that are used for mail services.

Table 1-5 Contents of the `/usr/lib` Directory Used for Mail Services

Name	Type	Description
`mail.local`	File	Mailer that delivers mail to mailboxes
`sendmail`	File	The routing program, also known as the mail transport agent

Several other files and directories are used by the mail services. These are shown in Table 1-6.

Table 1-6 Other Files Used for Mail Services

Name	Type	Description
`/var/spool/mqueue`	Directory	Where undelivered mail is stored
`/var/mail/mailbox1`, `/var/mail/mailbox2`	File	Mailboxes for delivered mail
`/etc/sendmailvars`	File	Stores macro and class definitions for lookup from `sendmail.cf`
`sendmailvars.org_dir`	Table	NIS+ version of `sendmailvars` file

Table 1-6 Other Files Used for Mail Services *(Continued)*

Name	Type	Description
`/usr/sbin/in.comsat`	File	Mail–notification daemon
`/usr/sbin/syslogd`	File	Error message logger, used by `sendmail`
`$OPENWINHOME/bin/mailtool`	File	Window–based interface to `sendmail`

Mail services are provided by a combination of these programs, which interact as shown by the simplified diagram in Figure 1-3.

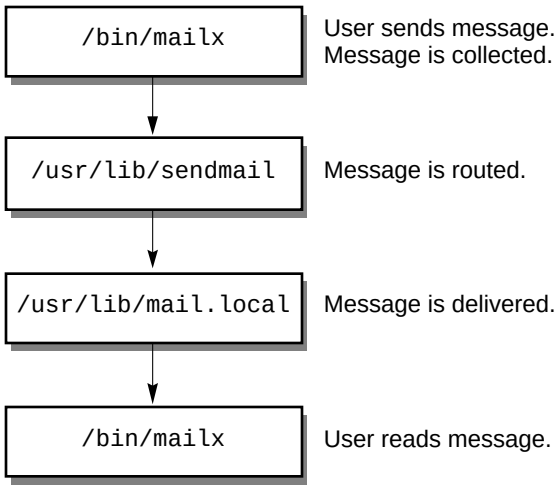


Figure 1-3 How Mail Programs Interact

Users send messages by using programs like `mailx` or `mailtool`. See the `mailx(1)` or `mailtool(1)` man pages for information about these programs.

The message is collected by the program that was used to generate it and is passed to the `sendmail` daemon. The `sendmail` daemon *parses* the addresses (divides them into identifiable segments) in the message, using information from the configuration file, `/etc/mail/sendmail.cf`, to determine network

name syntax, aliases, forwarding information, and network topology. Using this information, `sendmail` determines the route a message must take to get to a recipient.

The `sendmail` daemon passes the message to the appropriate system. The `/usr/lib/mail.local` program on the local system delivers the mail to the mailbox in the `/var/mail/`*username* directory of the recipient of the message.

The recipient is notified that mail has arrived, and retrieves it using `mail`, `mailx`, `mailtool`, or similar programs.

`sendmail` *Program*

The SunOS 5.x operating system uses the `sendmail` program as a mail router. `sendmail` is responsible for receiving and delivering electronic mail messages. It is an interface between mail–reading programs like `mail`, `mailx`, and `mailtool`, and mail–transport programs like `uucp`. The `sendmail` program controls email messages that users send, understands the recipients' addresses, chooses an appropriate delivery program, rewrites the addresses in a format that the delivery agent understands, reformats the mail headers as required, and finally passes the transformed message to the mail program for delivery.

Note – SunOS 2.4 and earlier releases included a binary called `sendmail.mx`. This program is now included in the sendmail program and the functionality is turned on by adding the `dns` flag to the hosts entry in `/etc/nsswitch.conf`. For more information, please read "Setting Up DNS Aliases Files" on page 46.

Figure 1-4 on page 19 shows how `sendmail` uses aliases. Programs that read mail, like `/usr/bin/mailx`, can have aliases of their own, which are expanded before the message reaches `sendmail`. The aliases for sendmail can come from a number of name space sources (local files, NIS or NIS+). The order of the lookup is determined by the `nsswitch.conf` file. See the `nsswitch.conf(4)` man page.

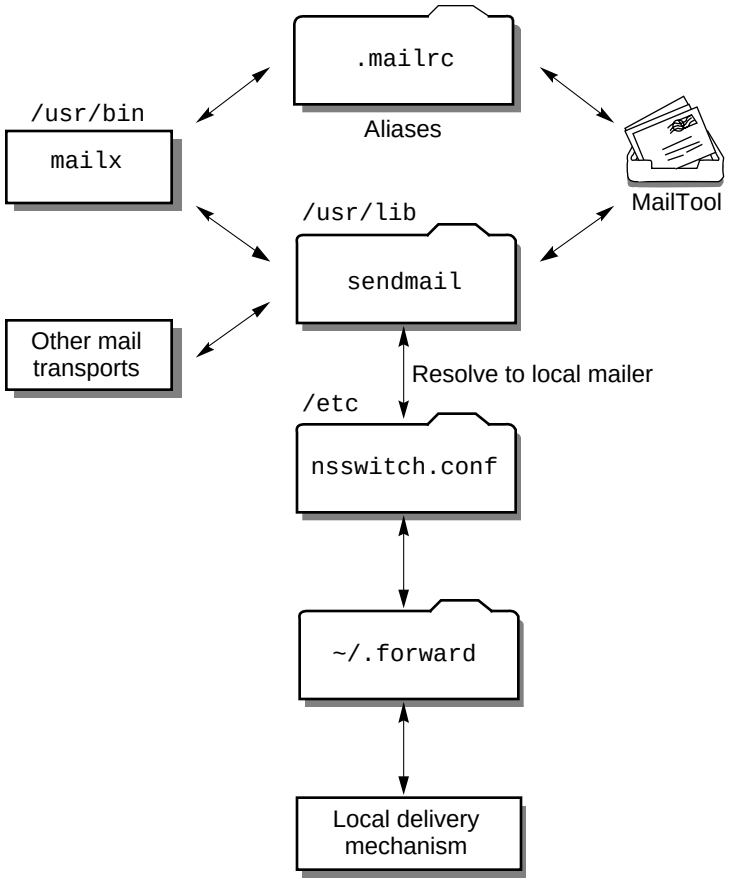


Figure 1-4 How `sendmail` Uses Aliases

`sendmail` *Configuration File*

A *configuration file* controls the way that `sendmail` performs its functions. The configuration file determines the choice of delivery agents, address rewriting rules, and the format of the mail header. A complete description of the file is presented in “sendmail Configuration File” on page 75.

1

The `sendmail` program uses the information from the `/etc/mail/sendmail.cf` file to perform its functions. Each system has a default `sendmail.cf` file installed in the `/etc/mail` directory. You do not need to edit or change the default configuration file for mail servers or mail clients. The only systems that require a customized configuration file are mail hosts and mail gateways.

The SunOS 5.x system provides two default configuration files in the `/etc/mail` directory:

- A configuration file named `main.cf` for the system (or systems) you designate as the mail host or a mail gateway
- A configuration file named `subsidiary.cf` (a duplicate copy of the default `sendmail.cf` file)

The configuration file you use on a system depends on the role the system plays in your mail service.

- For mail clients or mail servers, you do not need to do anything to set up or edit the default configuration file.
- To set up a mail host or gateway, copy the `main.cf` file and rename it `sendmail.cf` (in the `/etc/mail` directory). Then edit the `sendmail.cf` file to set the relay mailer and relay host parameters needed for your mail configuration.

The following list describes some configuration parameters you may want to change, depending on the requirements of your site:

- Time values specifies:
  - Read timeouts
  - How long a message remains undelivered in the queue before it is returned to the sender
- Delivery modes specifies how quickly mail will be delivered.
- Load limiting prevents wasted time during loaded periods by not attempting to deliver large messages, messages to many recipients, and messages to sites that have been down for a long time.
- Log level specifies what kinds of problems are logged.
- File modes set:
  - User ID (`setuid`) for `sendmail`
  - Temporary file modes

- `/etc/mail/aliases` permissions

`sendmail` *Configuration Table*

In response to two entries in the `sendmail.cf` file, the `sendmail` program can define macros and classes by looking up values in the `sendmailvars` configuration table. There are two such commands:

- Lines that begin with the `L` key letter are macro definitions, where the values assigned to the specified variable are obtained from the configuration table.
- Lines that begin with the `G` key letter are class definitions, where the values assigned to the specified variable are obtained from the configuration table.

The `L` command has the following syntax:

`L`*Xsearch_key*

For example: `Lmmaildomain`

In this case, the search key `maildomain` is used to look up a value in the configuration table to assign to the variable `m`. Most often the single-letter variable name is uppercase, but for internal variables (like `m` for the mail domain name) it is lowercase.

The `G` command has the following syntax:

`G`*Csearch_key*

For example: `GVuucp-list`

In this case, the search key `uucp-list` is used to look up a value in the configuration table to assign to the variable `V`.

In both cases, matching of the search key is case sensitive.

Both commands have counterparts for defining macros or classes within the `sendmail.cf` file, rather than the lookup table. `D` is the counterpart of `L`; `C` is the counterpart of `G`.

If NIS+ is used to administer the network, a global version of the sendmailvars information can be maintained. In addition to the NIS+ table or as an alternative, the data can be maintained in `/etc/mail/sendmailvars`. The order in which these sources are searched by `sendmail` is controlled by the

`sendmailvars` entry in the `/etc/nsswitch.conf` file. By default, the search order is `files nisplus`, which means `sendmail` attempts to look up information in the local file, before going to the NIS+ table.

Entries in an `/etc/mail/sendmailvars` file have the following format:

search_key [*value1 value2 value3...*]

The search key may be followed by a tab or several spaces; values are separated by a single space.

The NIS+ `sendmailvars` table has two columns: a key column and a value column. The value column can have one or more values, each separated by a space. For example:

Key Column	Value Column
`maildomain`	`eng.acme.com`
`uucp-list`	`acmemoon hugo comic`

Most mail variables should be defined in the NIS+ table. However, in special cases, systems can override the global setting for a variable by including it in their local `/etc/mail/sendmailvars` file.

`.forward` *Files*

Users can create a `.forward` file in their home directories that `sendmail` uses to temporarily redirect mail or send mail to a custom set of programs without bothering a system administrator. When troubleshooting mail problems, particularly problems with mail not being delivered to the expected address, always check the user's home directory for a `.forward` file.

A common mistake users make is to put a `.forward` file in the home directory of *host1* that forwards mail to user@host2. When the mail gets to *host2*, `sendmail` looks up `user` in the NIS or NIS+ aliases and sends the message back to user@host1, resulting in a loop, and more bounced mail.

How Mail Addressing Works

The path a mail message follows during delivery depends on the set up of the client system and the topology of the mail domain. Each additional level of mail hosts or mail domains can add one more round of alias resolution, but the routing process is basically the same on most hosts.

A client system can be set up to receive mail locally or a remote server can be selected to receive the mail. Receiving mail locally is known a running `sendmail` in local mode. Local is the default mode for all mail servers and some clients. If the client is mounting `/var/mail` from a server, then the client is running `sendmail` in remote mode.

Assuming that you are using the default rule set in the `sendmail.cf` file, the following examples show the route an email message takes.

On a mail client in remote mode, a mail message will go through the following routing process:

1. **Expand the mail alias, if possible, and restart the local routing process.**
   The mail address is expanded by looking up the mail alias in the name space, according to entry in `/etc/nsswitch.conf`, and substituting the new value, if one is found. This new alias is then checked again.
2. **If the address can not be expanded, forward it to the mail server.**
   If the mail address can not be `expanded`, then there could be a problem with the address or the address is not local. In both cases, the mail server will need to resolve the problem.
3. **If the expanded alias loops back to the original address, forward the mail to the mail server.**
   The process keeps a history of all of the lookups and if the original alias is generated again, the mail is forwarded to the mail server to resolve.

On the mail server or a mail client in local mode, a mail message will go through the following routing process:

1. **Expand the mail alias, if possible, and restart the local routing process.**
   The mail address is expanded by looking up the mail alias in the name space and substituting the new value, if one is found. This new alias is then checked again.
2. **If the mail is local; deliver it to `/usr/lib/mail.local`.**
   The mail will be delivered to a local mailbox.

3. **If the mail address includes a host in this mail domain, deliver the mail to that host.**
4. **If the address does not include a host in this domain, forward the mail to the mail host.**

The mail host will use the same routing process as the mail server, but the mail host will be able receive mail addressed to the domain name as well as to the hostname.

Planning Your Mail System

This section describes four basic types of mail configurations and briefly outlines the tasks required to set up each configuration. You may find this section useful if you need to set up a new mail system or if you are expanding an existing one. The configurations start with the most basic case (mail completely local, no connection to the outside world) and increase in complexity to a two-domain configuration with a mail gateway.

To set up a mail system, regardless of its configuration, you need these elements:

- A `sendmail.cf` configuration file on each system
- Alias files with an alias for each user to point to the place where mail is stored
- A mailbox to store (or spool) mail files for each user
- A `postmaster` alias for the person who administers mail services

How you set up the configuration file and the alias file and where you put the mailboxes depend on the configuration you choose.

As system administrator, you should decide on a policy for updating aliases and for forwarding mail messages. You might set up an `aliases` mailbox as a place for users to send requests for mail forwarding and for changes to their default mail alias. If your system uses NIS or NIS+, you can administer forwarding rather than forcing users to manage it themselves.

Local Mail Only

The simplest mail configuration, shown in Figure 1-5, is one mail host with two or more workstations connected to it. Mail is completely local. All of the clients store mail on their local disks and are acting as mail servers. Mail addresses are parsed using the `/etc/mail/aliases` files.

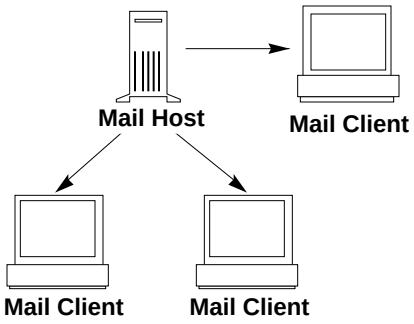


Figure 1-5 Local Mail Configuration

To set up this kind of mail configuration, you need:

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost` to the `/etc/hosts` file on the mail host; then if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Enough space in `/var/mail` on each mail client system to hold the mailboxes.

Local Mail in Remote Mode

In this configuration, each mail client mounts their mail from one mail server which provides mail spooling for client mailboxes. This server can also be the mail host. This configuration makes it easy to backup the mailboxes for each client.

To set up this kind of mail configuration, you need:

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost` to the `/etc/hosts` file on the mail host; then if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory
- Enough space in `/var/mail` on the mail server to hold the client mailboxes.

Local Mail and a Remote Connection

The most common mail configuration in a small network is shown in Figure 1-6. One system is the mail server, the mail host, and the mail gateway to the outside world. Mail is distributed using the `/etc/mail/aliases` files. No name service is required.

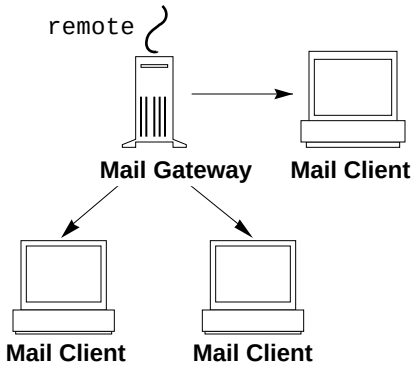


Figure 1-6 Local Mail Configuration With a UUCP Connection

To set up this kind of a mail configuration, assuming that the mail clients mount their mail files from `/var/mail` on the mail host, you need:

- The `main.cf` file on the mail gateway (you must edit the file to select a major relay mailer)

- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- A server designated as the mail host (add `mailhost` to the `/etc/hosts` file on the mail host; if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+)
- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory when mailboxes are located on the mail host
- Enough space in `/var/mail` on the mail server to hold the client mailboxes.

Two Domains and a Gateway

The mail configuration shown in Figure 1-7 has two domains and a mail gateway. In this configuration, the mail server, the mail host, and the mail gateway (or gateways) for each domain are likely to be different systems. To make the process of administering and distributing mail easier, a name service is used.

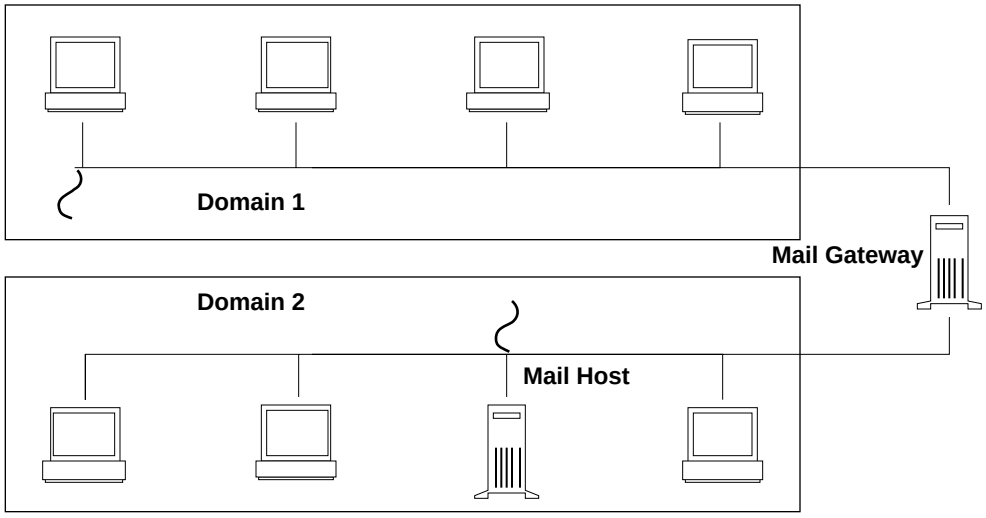


Figure 1-7 Two Domains and a Gateway

To set up this kind of a mail configuration, assuming that the mail clients mount their mail files from `/var/mail` on the mail host, you need:

- Complex gateway systems usually need a customized `sendmail.cf` file with special rules added
- The `main.cf` file on the mail gateway (you must edit the file to select a major relay mailer)
- A server designated as the mail host (add `mailhost` to the `/etc/hosts` file on the mail host; if you are not running NIS or NIS+, add the mail host IP address line to the `/etc/hosts` file of all mail clients)
- Matching `/etc/mail/aliases` files on any system that has a local mailbox (unless you are running NIS or NIS+).
- An alias entry for each user, to point to where the mail is stored, in `mail_aliases.org_dir` for NIS+ or the aliases map for NIS
- The default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)

- Entries in each mail client's `/etc/vfstab` file or `/etc/auto_direct` (if autofs is used) to mount the `/var/mail` directory when mailboxes are located on the mail host
- Enough space in `/var/mail` on the mail server to hold the client mailboxes.

1

Setting Up and Administering Mail Services

2

This chapter describes how to set up and administer mail services.

If you are not familiar with administering mail services, read Chapter , “Understanding Mail Services,” for an introduction to the terminology and structure of the mail services and for descriptions of several mail service configurations.

Use the following table to find the page where the instructions for specific tasks.

Setting Up Mail Services

You can set up a mail service relatively easily if your site does not provide connections to electronic mail (email) services outside your company or if your company is in a single domain.

2

Chapter 3, “Customizing sendmail Configuration Files,” contains information about how to create more complicated configuration files.

Mail requires two types of configurations for local mail and two more for communication with networks outside of your domain. These configurations can be combined on the same system or provided by separate systems. You need to set up systems on your site to perform the functions described in Table 2-1.

Table 2-1 Mail Configurations

Configuration	Description
Mail client	Mail clients are users who have mailboxes on a mail server.
Mail server	The mail server stores mailboxes in the `/var/mail` directory.
Mail host	You need at least one mail host. The mail host resolves difficult email addresses and reroutes mail within your domain.
Mail gateway	A mail gateway is a connection between different networks outside your domain or between differing communications networks. You must add rules to the `sendmail.cf` file to set up a gateway. See Chapter 3, “Customizing sendmail Configuration Files,” for information about adding rules. If you have to set up a mail gateway, you should find a gateway configuration file that is close to what you need and modify it to fit your situation.

Before you begin to set up your mail service, choose the systems to act as mail servers, mail hosts, and mail gateways. You should also make a list of all the mail clients for which you will be providing service and include the location of their mailboxes. This list will help you when you are ready to create mail aliases for your users. See Chapter 1, “Understanding Mail Services,” for more information about the function each of these systems provides. For your convenience, guidelines about which systems are good candidates for mail server, mail host, and mail gateways are repeated in the following sections.

To simplify the setup instructions, this chapter tells you what you need to do to set up individual mail servers, mail hosts, mail clients, and relay hosts. If a system in your mail services configuration is acting in more than one capacity, follow the appropriate instructions for each type of system. For example, if your mail host and mail server functions are on the same system, follow the directions for setting up that system as a mail host and then follow the directions for setting up the same system as a mail server.

Note – The following procedures for setting up a mail server and mail client apply when mailboxes are NFS-mounted. However, mailboxes typically are maintained in locally mounted `/var/mail` directories — in which case the following procedures are not needed.

Setting Up Mail Security

By default, security permissions on a `/var/mail` directory allow read, write, and execute access to the owner, members of groups to which the owner belongs, and all others. On Solaris 2.x mail servers, you can make a `/var/mail` directory more secure by changing its default permissions to allow only read and write access to anyone outside the owner's groups, as long no SunOS 4.1.x mail clients are connected. For more information about changing directory permissions, refer to *System Administration Guide, Volume I.*

▼ How to Set Up a Mail Server

There are no special steps required to set up a mail server that is only serving the mail for local users. The user must have an entry in the password file or in the namespace, and the user should have a local home directory (so that `~/.forward` can be checked) for mail to be delivered. This is why home directory servers are often set up as the mail server.

The mail server may route all mail for many mail clients. The only resource requirement for this type of mail server is that it has adequate spooling space for client mailboxes. The `/var/mail` directory must be made available for remote mounting.

For this task, you will check the `/etc/dfs/dfstab` file to be sure the `/var` directory is exported.

1. **Type** `share` **and press Return.**
   If the `/var` directory is shared, you are done. If the `/var` directory is not exported, continue with the next step.
2. **Type** `share -F nfs /var/mail` **and press Return.**
3. **To permanently share the file system, edit** `/etc/dfs/dfstab` **and add the command line used in Step 2.**

Note – The `mail.local` program automatically creates mailboxes in the `/var/mail` directory the first time a message is delivered. You do not need to create individual mailboxes for your mail clients.

▼ How to Set Up a Mail Client

A mail client is a user of mail services, with a mailbox on a mail server, and a mail alias in the `/etc/mail/aliases` file that points to the location of the mailbox.

1. **Become root on the mail client's system.**
2. **Make sure that there is a `/var/mail` mount point on the mail client's system.**
3. **Mount the `/var/mail` directory from the mail server.**
   The mail directory may be automatically mounted or mounted at boot time.

   a. **To mount `/var/mail` automatically, edit `/etc/auto_direct` and add an entry like the one below:**

```
/var/mail -rw,hard,noac server:/var/mail
```

   b. **To mount `/var/mail` at boot time, edit the `/etc/vfstab` file and add an entry for the `/var/mail` directory on the mail server, mounting it on the local `/var/mail` directory.**

```
server:/var/mail - /var/mail nfs - no rw,hard,noac
```

   The client's mailbox will be automatically mounted any time the system is rebooted. Type `mountall` to mount the client mailbox until the system is rebooted.

Caution – The `noac` option must be included when mounting mail from a NFS server to allow mailbox locking and access to work properly.

4. **Use the Administration Tool to edit the `/etc/hosts` file and add an entry for the mail server.**
   This step is not required if you are using a name service.

5. **Add an entry for the client to one of the alias files.**
   See "Creating Mail Aliases" on page 37 for information about how to create mail aliases for different kinds of mail configurations.

Note – The `mail.local` program automatically creates mailboxes in the `/var/mail` directory the first time a message is delivered. You do not need to create individual mailboxes for your mail clients.

▼ How to Set Up a Mail Host

A mail host resolves email addresses and reroutes mail within your domain. A good candidate for a mail host is a system that connects your systems to the outside world or to a parent domain.

1. **Become root on the mail host system.**

2. **Use the Administration Tool to edit the `/etc/hosts` file.**
   Add the word `mailhost` after the IP address and system name of the mail host system. The system is designated as a mail host.

3. **Create an entry for the new mail host in one of the hosts files.**
   If you are using NIS or NIS+, add an entry including a host alias called `mailhost` to the host entry for the new mail host.

   If you are not using NIS or NIS+, you must create an entry in `/etc/hosts` for each system on the network. The entry should use this format: *IP address mailhost_name* `mailhost`

4. **Type `cp /etc/mail/main.cf /etc/mail/sendmail.cf` and press Return.**
   This copies and renames the `/etc/mail/main.cf` file.

5. **Reboot the mail host and test your mail configuration.**
   See "Testing the Mail Configuration" on page 48 for information.

2

▼ How to Set Up a Mail Gateway

A mail gateway manages communication with networks outside of your domain. The mailer on the sending mail gateway may match the mailer on the receiving system.

A good candidate for a mail gateway is a system attached to Ethernet and phone lines or a system configured as a router to the Internet. You may want to configure the mail host or another system as mail gateway. You may choose to configure more than one mail gateway for your domain. If you have UUCP connections, you should configure the system (or systems) with UUCP connections as the mail gateway.

1. **Become root on the mail gateway.**
2. **Type** `cp /etc/mail/main.cf /etc/mail/sendmail.cf` **and press Return.**
   This command copies and renames the `main.cf` file.
3. **Edit the** `/etc/mail/sendmail.cf` **file and make the following changes:**
   a. **Only if your relay mailer is not UUCP, change the default entry** `DMsmartuucp` **to the entry that is appropriate for your relay mailer.**
   Available mailers are `smartuucp` (the default), `ddn`, `ether`, and `uucp`. If your relay mailer is UUCP, you do not need to change this entry.

   You can specify a different relay mailer for each mail gateway (if appropriate). You can define rule sets for other relay mailers in the `main.cf` file. See “Mailers” on page 3 for a description of each of the default relay mailers.

   b. **In the entry** `DR ddn-gateway`**, replace** `ddn-gateway` **with the name of your mail relay.**
   The `DR` entry defines the mail relay.

   c. **In the entry** `CR ddn-gateway`**, replace** `ddn-gateway` **with the name of your mail relay.**
   The `CR` entry defines the class of the mail relay. You can designate one or more hosts as a member of this class.

   d. **(Optional) Add a** `Dmmaildomain` **or** `Lmmaildomain` **entry to define the mail domain name to be used for mail delivery.**
   The `m` macro defines the mail domain name. If the macro is not defined, the naming service domain name is used with the first component stripped off. For example, `ecd.east.acme.com` becomes

`east.acme.com`. If you use the `L` command, `sendmail` will look up the name to use in the `sendmailvars` table, using `maildomain` as the search key.

e. **Save the edits.**

4. **Reboot the mail gateway and test your mail configuration.**
   See "Testing the Mail Configuration" on page 48 for information.

Creating Mail Aliases

You can use the `aliasadm` command to create mail aliases for a user. Mail aliases must be unique within the domain. This section tells you how to use command lines to search the mail aliases table for aliases, and to create mail aliases for NIS+, NIS, DNS, or on the local system.

Or you can use the Administration Tool's Database Manager application to perform these tasks on the aliases database.

Which type of file to use depends on who will be using the alias and who needs to be able to change the alias. Each type of alias file has unique format requirements. Each of these will be defined in the following sections.

`.mailrc` *Aliases*

Aliases listed in a `.mailrc` file are accessible only by the user who owns the file. This allows users to establish an alias file they control and that is usable only by its owner. Aliases in a `.mailrc` file adhere to the following format:

```
alias aliasname value value value ...
```

where *aliasname* is the name the user will use when sending mail, and *value* is a valid email address.

If a user establishes a personal alias for `scott` that does not match the email address for `scott` in the namespace, mail will get routed to the wrong person when other people try to reply to mail generated by that user. The only workaround is to use any of the other aliasing mechanisms.

/etc/mail/aliases

Any alias established in the /etc/mail/aliases file can be used by any user who knows the name of the alias and the host name of the system that contains the file. Distribution list formats in a local /etc/mail/aliases file adhere to the following format:

```
aliasname: value,value,value...
```

where *aliasname* is the name the user will use when sending mail to this alias and *value* is a valid email address.

The aliases in the /etc/mail/aliases file are stored in text form. When you edit the /etc/mail/aliases file, run the newaliases program to recompile the database and make the aliases available in binary form to the sendmail program. Or you can use Administration Tool's Database Manager to administer the mail aliases stored in local /etc files.

Normally, this file can be edited only by the root user. If using the Administration Tool, then all users in group 14, which is the sysadmin group, will be able to change the local file. Another option is to create an entry like:

```
aliasname: :include:/path/aliasfile
```

where *aliasname* is the name the user will use when sending mail and */path/aliasfile* is the full path to the file that includes the alias list. The alias file should include email entries, one entry on each line, and no other notations:

```
user1@host1
user2@host2
```

The permissions on this file can be changed to allow multiple users to be able to change the alias, without giving out root access or permissions to change data with the Administration Tool.

It is possible to describe additional mail files in `/etc/mail/aliases` to keep a log or a backup copy. The entry below will store all mail sent to *aliasname* in *filename*.

```
aliasname: /home/backup/filename
```

Warning – All mail files must be writable by `daemon`. To allow other users to deliver mail to this file, the permissions should be set so that the file is owned by a specific user, the group owner must be `daemon`, and the file permissions should be at least `0620`.

It is also possible to route the mail to another process. The entry below will store a copy of the mail message in *filename* and will print a copy.

```
aliasname: "|tee -a /home/backup/filename |lp"
```

NIS Aliases Map

Entries included in the NIS aliases map can be used by all users in the local domain. The `sendmail` program can use the NIS aliases map instead of the local `/etc/mail/aliases` files to determine mailing addresses. See the `nsswitch.conf(4)` man page for more information.

Aliases in the NIS `aliases` map adhere to the following format:

```
aliasname: value,value,value...
```

where *aliasname* is the name the user will use when sending mail and *value* is a valid email address.

The NIS aliases map should contain entries for all mail clients. In general, these entries can only be changed by the root user on the NIS master. This type of alias may not be a good choice for aliases that are constantly changing, but can be useful if the alias points to another alias file; as in this syntax example:

```
aliasname: aliasname@host
```

where *aliasname* is the name the user will use when sending mail and *host* is the hostname for the server which contains an `/etc/mail/alias` file.

NIS+ mail_aliases Table

The NIS+ mail_aliases table contains the names by which a system or person is known in the local domain. The `sendmail` program can use the NIS+ mail_aliases table instead of the local `/etc/mail/aliases` files to determine mailing addresses. See the `aliasadm(1M)` and `nsswitch.conf(4)` man pages for more information.

Aliases in the NIS+ mail_aliases table adhere to the following format:

```
alias:          expansion               [options  #  "comments"]
```

The four columns are described in Table 2-2.

Table 2-2 Columns in the NIS+ mail_aliases Table

Column	Description
alias	The name of the alias
expansion	The value of the alias or a list of aliases as it would appear in a `sendmail /etc/mail/aliases` file
options	Reserved for future use
comments	Comments about an individual alias

The NIS+ mail_aliases table should contain entries for all mail clients. You can list, create, modify, and delete entries in the NIS+ aliases table with the `aliasadm` command. Or you can use Administration Tool's Database Manager to administer NIS+ mail aliases.

If you are creating a new NIS+ aliases table, you must initialize the table before you create the entries. If the table exists, no initialization is needed. See "How to Add Aliases to a NIS+ mail_aliases Table From the Command Line" on page 41 for information about how to create an NIS+ mail_aliases table.

To use the `aliasadm` command, you must be a member of the NIS+ group that owns the aliases table or the person who created the table.

▼ How to List the Contents of an NIS+ Aliases Table

To use the `aliasadm` command, you must be either root, a member of the NIS+ group that owns the mail_aliases table, or the person who created the table.

To List the Entire Contents of the NIS+ mail_aliases Table

♦ **Type `aliasadm -l` and press Return.**
This lists the contents of the aliases table in alphabetic order by alias.

Note – If you have a large aliases table, listing the entire contents can take some time. If you are searching for a specific entry, pipe the output through the `grep` command (`aliasadm -l | grep` *entry*) so that you can use the `grep` search capability to find specific entries.

To List Individual Entries in the NIS+ mail_aliases Table

♦ **Type `aliasadm -m` *alias* and press Return.**
The alias entry is listed.

```
# aliasadm -m ignatz
ignatz: ignatz@saturn #Alias for Iggy Ignatz
```

Note – The `aliasadm -m` option matches only the complete alias name. It does not match partial strings. You cannot use metacharacters (like * and ?) with the `aliasadm -m` option. If you are interested in partial matches, type `aliasadm -l | grep` *partial-string* and press Return.

▼ How to Add Aliases to a NIS+ mail_aliases Table From the Command Line

If you are creating a completely new NIS+ mail_aliases table, you first must initiate the NIS+ table.

To Initiate an NIS+ Table

♦ **Type `aliasadm -I` and press Return.**

To Add Aliases to an NIS+ mail_aliases Table From the Command Line

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**
2. **Become root on any system.**
3. **For each alias, type** `aliasadm -a` *alias expanded_alias* [*options comments*] **and press Return.**
   This adds the aliases to the NIS+ aliases table.

```
# aliasadm -a iggy iggy.ignatz@saturn "Iggy Ignatz"
```

4. **Type** `aliasadm -m` *alias* **and press Return.**
   This displays the entry you created.
5. **Check the entry to be sure it is correct.**

▼ How to Add Entries by Editing an NIS+ mail_aliases Table

If you are adding more than two or three aliases, you may want to edit the NIS+ table directly.

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**
2. **Become root on any system.**
3. **Type** `aliasadm -e` **and press Return.**
   The aliases table is displayed using the editor set with the `$EDITOR` environment variable. If the variable is not set, the `vi` editor is used.
4. **Type each alias on a separate line, using these formats:**
   a. **Enter the aliases in any order, at any place in the table.**
      The order is not important to the NIS+ mail_aliases table. The `aliasadm -l` command sorts the list and displays them in alphabetical order.
   b. **Use the format** *alias*`:` *expanded_alias* `#` `["`*option*`"# "`*comments*`"]`
      If you leave the option column blank, enter an empty pair of quotation marks (`""`) and then add the comments.
   c. **End each line by pressing Return.**

5. **Check that the entries are correct.**
6. **Save the changes.**

▼ How to Change Entries in an NIS+ mail_aliases Table

1. **Become root on any system.**
2. **Type** `aliasadm -m` *alias* **and press Return.**
   The information for the alias is displayed.
3. **Type** `aliasadm -c` *alias expanded_alias* [*options comments*] **and press Return.**
   The alias is changed using the new information you provide.
4. **Type** `aliasadm -m` *alias* **and press Return.**
   The entry you created is displayed.
5. **Check the entry to be sure it is correct.**

▼ How to Delete Entries From a NIS+ mail_aliases Table

1. **Become root on any system.**
2. **Type** `aliasadm -d` *alias* **and press Return.**
   The alias is deleted from the NIS+ mail_aliases table.

Setting Up NIS mail.aliases Map

The `/etc/mail/aliases` file on an NIS master contains all names by which a system or person is known. The NIS master is searched if there is no match in the local `/etc/mail/aliases` files. The `sendmail` program uses the NIS master file to determine mailing addresses. See the `aliases(4)` man page.

You can either edit the file on each system or edit the file on one system and copy it to each of the other systems.

Aliases are in the following form:

```
name: name1, name2, ...
```

You can use aliases for local names or domains. For example, an alias entry for user `fred` who has a mailbox on the system `saturn` and who is in the domain `planets` would have this entry in the `/etc/mail/aliases` file:

```
fred: fred@planets
```

▼ How to Set Up NIS mail.aliases Map

1. **Compile a list of each of your mail clients, the locations of their mailboxes, and the names of the mail server systems.**
2. **Become root on the NIS master server.**
3. **Edit the `/etc/mail/aliases` file, and make the following entries:**
   a. **Add an entry for each mail client.**
   b. **Change the entry `Postmaster: root` to the mail address of the person who is designated as postmaster.**
   See "Setting Up the Postmaster Alias" on page 47 for more information.
   c. **If you have created a mailbox for administration of a mail server, create an entry for `root:`*mailbox@mailserver*.**
   d. **Save the changes.**
4. **Edit the `/etc/hosts` file on the NIS master server and create an entry for each mail server.**
5. **Type `cd /var/yp` and press Return.**
6. **Type `make` and press Return.**
   The changes in the `/etc/hosts` and `/etc/mail/aliases` files are propagated to NIS slave systems. It takes a few minutes, at most, for the aliases to take effect.

Setting Up Local Mail Aliases Files

The `/etc/mail/aliases` file on a local system contains all names by which a system or person is known. The `sendmail` program uses this file to determine mailing addresses. See the `aliases(4)` man page.

If your network is not running a name service, the `/etc/mail/aliases` file of each system should contain entries for all mail clients. You can either edit the file on each system or edit the file on one system and copy it to each of the other systems.

Aliases are of the form:

```
name: name1, name2, ...
```

You can create aliases for only local names—a current host name or no host name. For example, an alias entry for user `ignatz` who has a mailbox on the system `saturn` would have this entry in the `/etc/mail/aliases` file:

```
ignatz: ignatz@saturn
```

It is a good idea to create an administrative account for each mail server. You do this by assigning root a mailbox on the mail server and adding an entry to the `/etc/mail/aliases` file for root. For example, if the system `saturn` is a mailbox server, add the entry `root: sysadmin@saturn` to the `/etc/mail/aliases` file.

▼ How to Set Up Local Mail Aliases Files

1. **Compile a list of each of your mail clients and the locations of their mailboxes.**
2. **Become root on the mail server.**
3. **Edit the `/etc/mail/aliases` file and make the following entries:**
   a. **Add an entry for each mail client.**
   b. **Change the entry `Postmaster: root` to the mail address of the person who is designated as postmaster.**
      See “Setting Up the Postmaster Alias” on page 47 for more information.
   c. **If you have created a mailbox for administration of a mail server, create an entry for `root:` *mailbox@mailserver*.**
   d. **Save the changes.**

4. **Type** `newaliases` **and press Return.**
   This creates an `alias` file in binary form that `sendmail` can use. The file is stored in the `/etc/mail/aliases.dir` and `/etc/mail/aliases.pag` files.

5. **Copy the** `/etc/mail/aliases`**, the** `/etc/mail/aliases.dir`**, and** `/etc/mail/aliases.pag` **files to each of the other systems.**
   When you copy all three files, you do not need to run the `newaliases` command on each of the other systems.

   You can copy the files by using the `rcp` or `rdist` commands or by using a script that you create for this purpose. Remember that you must update all the `/etc/mail/aliases` files each time you add or remove a mail client.

Setting Up DNS Aliases Files

The DNS name service does not support aliases for individuals. It does support aliases for hosts or domains using *mail exchange* (MX) *records* and cname records. You can specify host names, domain names, or both in the DNS database. Domain names can contain wildcards. For example, `*.acme.com` is an acceptable domain name. See *NIS+ and DNS Setup and Configuration Guide* for more information about administering DNS.

▼ How to Use DNS with `sendmail`

1. **Edit the** `/etc/nsswitch.conf` **file and make sure that the hosts entry includes the** `dns` **flag.**
   The host entry must include the `dns` flag in order for the DNS host aliases to be used.

2. **Check for a** `mailhost` **entry.**
   Make sure there is an entry for `mailhost` in the DNS database.

Setting Up the Postmaster Alias

Every system should be able to send mail to a `postmaster` mailbox. You can create an NIS or NIS+ alias for `postmaster` or create one in each local `/etc/mail/aliases` file. Here is the default `/etc/mail/aliases` entry:

```
# Following alias is required by the mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's
# mail problems.
Postmaster: root
```

To create the `postmaster` alias, edit each system's `/etc/mail/aliases` file and change `root` to the mail address of the person who will act as postmaster.

You may want to create a separate mailbox for the postmaster to keep postmaster mail separate from personal mail. If you create a separate mailbox, use the mailbox address instead of the postmaster's mail address when you edit the `/etc/mail/aliases` files.

▼ How to Create a Separate Mailbox for `postmaster`

1. **Create a user account for the person designated as `postmaster` and put an asterisk (*) in the password field.**
2. **Once mail has been delivered, type `mail -f` *postmaster* and press Return.**
   The `mail` program will be able to read and write to the mailbox name.

▼ How to Add the `postmaster` Mailbox to the Aliases

1. **Become root and edit the `/etc/mail/aliases` file on each system.**
   If your network does not run NIS or NIS+, edit the `/etc/mail/aliases` file.
2. **Change the `postmaster` alias from `root` to `Postmaster:` *postmastermailbox@postmasterhost* and save the changes.**
3. **On the postmaster's local system create an entry in the `/etc/mail/aliases` file that defines the name of the alias (`sysadmin`, for example) and includes the path to the local mailbox.**
4. **Type `newaliases` and press Return.**

Or you could change the `Postmaster:` entry in the `aliases` file to a `Postmaster`: */usr/somewhere/somefile* entry.

Testing the Mail Configuration

When you have set up all the systems in your mail configuration, use the suggestions in this section to test the setup to be sure mail messages can be sent and received.

▼ How to Test the Mail Configuration

1. **Reboot any system for which you have changed a configuration file.**
2. **Send test messages from each system by typing** `/usr/lib/sendmail -v` *names* `</dev/null` **and press Return.**
   Specify a recipient's email address in place of the *names* variable.

   This command sends a null message to the specified recipient and displays messages while it runs.
3. **Run these tests:**
   a. **Send mail to yourself or other people on the local system by addressing the message to a regular user name.**
   b. **If you are on Ethernet, send mail to someone on another system.**
   Do this in three directions: from the main system to a client system, from a client system to the main system, and from a client system to another client system.
   c. **If you have a mail gateway, send mail to another domain from the mail host to ensure that the relay mailer and host are configured properly.**
   d. **If you have set up a UUCP connection on your phone line to another host, send mail to someone at that host and have that person send mail back or call you when the message is received.**
   e. **Ask someone to send mail to you over the UUCP connection.**
   The `sendmail` program cannot tell whether the message gets through, because it hands the message to UUCP for delivery.
   f. **Send a message to** `postmaster` **on different systems and make sure that it comes to your postmaster's mailbox.**

Administering the Mail Configuration

This section describes how to keep the mail service running smoothly.

Duties of the Postmaster

As postmaster your responsibilities for administering mail include the following tasks:

- Check the mail queues to be sure mail is flowing in and out.
- Check any downed systems where mail is backing up. If the system is not needed, delete it from the mail service or bring the system up to keep mail moving.
- Fix personal aliases, as requested.
- Administer alias databases as people move in and out of the domain.
- Set up temporary forwarding files.
- Contact owners of mailing lists and help them fix mailing list problems.
- Go through `postmaster` mail daily and look for problems, like broken `.forward` files and mail alias loops.
- Answer questions outside the company.
- Truncate log files periodically.

Mail Queue

Under high-load or temporary-failure conditions, `sendmail` puts a message into a job queue instead of delivering it immediately. The mail queue usually is processed automatically, but sometimes you may have to intervene. For example, if a major host is down for a period of time, the queue could become clogged. Although `sendmail` ought to recover gracefully when the host comes up, you could find performance unacceptably bad in the meantime.

Format of Queue Files

The `sendmail` program stores temporary queue files in the queue file `/var/spool/mqueue`. All such queue files have the form *xfAA99999*, where *AA99999* is the ID for the file and *x* is the type. Table 2-3 shows the types of queue files.

Table 2-3 Types of Queue Files

Type	Description
`d`	Data file. The message body (excluding the header) is kept in this file.
`q`	Queue control file. This file contains the information needed to process the job.
`t`	A temporary file. This file is an image of the `qf` file when it is being rebuilt. When the rebuild is complete, the file is renamed `qf`.
`x`	Transcript file that shows everything that happens during that session.

A `qf` file contains a series of lines each beginning with a code letter. Not all of these are included with each message, but a complete list is shown in Table 2-4.

Table 2-4 Codes for the `qf` File

Code	Description
`$`	A macro definition. The values of certain macros (currently `$r` and `$s`) are passed through to the queue run phase.
`B`	The body type This line defines the body type. Legal values are 7BIT and 8BITMIME.
`C`	The controlling address. The syntax is *localuser:aliasname*. Recipient addresses following this line will be flagged so that deliveries will be run as the *localuser* (a user name from the password file or database); *aliasname* is the name of the alias that expanded to this address (for printing messages).
`D`	The name of the data file. There may only be one of these lines.
`E`	The error recipient name. Error messages are sent to this user instead of the sender. This line is optional.
`F`	Flag bits. This can be r, indicating that this is a response message and w, indicating that a warning message has been sent announcing that the mail has been delayed.

Table 2-4 Codes for the `qf` File *(Continued)*

Code	Description
`H`	A header definition. There may be any number of these lines. The order is important: it represents the order in the final message. The syntax is the same as header definitions in the configuration file.
`L`	Information to compute the content length.
`M`	A message line, which is printed by using `sendmail` with the `–bp` flag and is generally used to store status information. It can contain any text.
`P`	The current message priority, which is used to order the queue. The higher the number, the lower the priority. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.
`R`	A recipient name. There will be one line for each recipient. The recipient name will normally be completely aliased, but its aliases are redone when the job is processed. The recipient name must be at the end of the `qf` file.
`S`	The sender name. There may be only one of these lines.
`T`	The job creation or submission time in seconds, which is used to compute when the job times out.

See Chapter 3, “Customizing sendmail Configuration Files,” for more information.

The queue is automatically run at the interval specified in the `sendmail.cf` file (the default is every hour). The queue is read and sorted, and then `sendmail` tries to process all jobs in order. The `sendmail` program first checks to see if a job is locked. If the job is locked, `sendmail` ignores the job; if the job is not locked, `sendmail` processes it.

If a major host goes down for several days, the queue may become prohibitively large and `sendmail` will spend time sorting the queue. You can fix this by moving the queue to a temporary place and creating a new queue. You can run the old queue later when the host is returned to service.

▼ How to Print the Queue

You can print the contents of the queue with `mailq.`This command is equivalent to specifying the `-bp` flag to `sendmail`.

- **Type** `/usr/bin/mailq | more` **and press Return.**
  A list of the queue IDs, the size of the message, the date the message entered the queue, the message status, and the sender and recipients are displayed.

How to Force the Queue

- **Type** `/usr/lib/sendmail -q -v` **and press Return.**
  This will force the processing of the queue and will display progress of the jobs as the queue is cleared.

How to Run a Subset of the Mail Queue

- **Type** `/usr/lib/sendmail -qR`*string* **and press Return.**
  You can run a subset of the queue at any time with the `-qR`*string* (run queue where any recipient name matches *string*) or with `-qI`*nnnn* (run just one message with queue ID *nnnn*).

This example processes everything in the queue for recipient, `wnj`.

```
# /usr/lib/sendmail -qRwnj
```

How to Move the Queue

1. **Become root on the mail host.**
2. **Type** `/etc/init.d/sendmail stop` **and press Return.**
   This kills the old `sendmail` daemon to keep it from trying to process the old queue directory.
3. **Type** `cd /var/spool` **and press Return.**
4. **Type** `mv mqueue omqueue; mkdir mqueue` **and press Return.**
   This moves the directory, `mqueue`, and all its contents to the omqueue directory and then creates a new empty `mqueue` directory.
5. **Type** `chmod 755 mqueue; chown daemon.daemon mqueue;` **and press Return.**
   These commands set the permissions of the directory to read/write/execute by owner, and read/execute by group and others; these commands also set the owner and group to daemon.

6. **Type** `/etc/init.d/sendmail start` **and press Return.**
   This starts a new `sendmail` daemon.

▼ How to Run the Old Mail Queue

1. **Type** `/usr/lib/sendmail -oQ/var/spool/omqueue -q` **and press Return.**
   The `-oQ` flag specifies an alternate queue directory and the `-q` flag says to run every job in the queue. Use the `-v` flag if you want to see the verbose output displayed on the screen.

2. **When the queue is finally emptied, type** `rmdir /var/spool/omqueue` **and press Return.**
   This removes the empty directory.

System Log

The mail services log most errors using the `syslogd` program. The default is for `syslogd` to send messages to the `loghost`.

You can define a system called `loghost` in the `/etc/hosts` file to hold all logs for an entire NIS domain. The system log is supported by the `syslogd` program. You specify a `loghost` in `/etc/hosts`. If no `loghost` is specified, then error messages from `syslogd` are not reported.

Table 2-5 shows the default `/etc/syslog.conf` file:

Table 2-5 The Default `/etc/syslog.conf` File

```
#ident "@(#)syslog.conf   1.3       93/12/09 SMI"  /* SunOS 5.0 */
#
# Copyright (c) 1994 by Sun Microsystems, Inc.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (‘’) names
# that match m4 reserved words. Also, within ifdef’s, arguments
# containing commas must be quoted.
#
# Note: Have to exclude user from most lines so that user.alert
#       and user.emerg are not included, because old sendmails
#       have no 4.2BSD based systems doing network logging, you
#       can remove all the special cases for "user" logging.
#
*.err;kern.debug;auth.notice;user.nonedev/console
*.err;kern.debug;daemon,auth.notice;mail.crit;user.none
/var/adm/messages

*.alert;kern.err;daemon.err;user.noneoperator
*.alert;user.noneroot

*.emerg;user.none                                *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice         ifdef(‘LOGHOST’, /var/log/authlog, @loghost)

mail.debug           ifdef(‘LOGHOST’, /var/log/syslog, @loghost)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(‘LOGHOST’, ,
user.err/dev/console
user.err/var/adm/messages
user.alert‘root, operator’
user.emert*
)
```

You can change the default configuration by editing the `/etc/syslog.conf` file. The syslog daemon must be restarted for any changes to take effect. These selections can be added to the file to gather information about mail:

- mail.alert - messages about conditions that should be fixed now
- mail.crit - critical messages
- mail.warning - warning messages
- mail.notice - messages that are not errors, but may need attention
- mail.info - informational messages
- mail.debug - debugging messages.

The entry below will send a copy of all critical, informational and debug messages to `/var/log/syslog`.

```
mail.crit;mail.info;mail.debug    /var/log/syslog
```

Each line in the system log contains a time stamp, the name of the system that generated it, and a message. The `syslog` file can log a large amount of information.

The log is arranged as a succession of levels. At the lowest level, only unusual occurrences are logged. At the highest level, even the most mundane and uninteresting events are recorded. As a convention, log levels under 10 are considered “useful.” Log levels higher than 10 are usually used for debugging. See *System Administration Guide, Volume I* for information about `loghost` and the `syslogd` program.

Troubleshooting Tips

This section provides some tips and tools that you can use for troubleshooting problems with the mail services.

▼ How to Check Aliases

To verify aliases and whether mail can be delivered to a given recipient:

♦ **Type** `/usr/lib/sendmail -v -bv` *recipient* **and press Return.**
The command displays the aliases and identifies the final address as deliverable or not.

Here is an example of the output:

```
% /usr/lib/sendmail -v -bv shamira@raks
shamira... aliased to   mwong
mwong... aliased to              shamira@raks
shamira@raks... deliverable
%
```

You should take extra care to avoid loops and inconsistent databases when both local and domain-wide aliases are used. Be especially careful when you move a user from one system to another to avoid creating alias loops.

▼ How to Test the `sendmail` Rule Sets

1. **Type** `/usr/lib/sendmail -bt` **and press Return.**
   Information is displayed.
2. **At the last prompt (>) type a** `3,0` **(zero) and the mail address you want to test.**
3. **Type Control-d to end the session.**

Here is an example of the output:

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 shimara@raks
rewrite: ruleset  3   input: shimara @ raks
rewrite: ruleset  6   input: shimara <@ raks>
rewrite: ruleset  6 returns: shimara <@ raks>
rewrite: ruleset  3 returns: shimara <@ raks>
rewrite: ruleset  0   input: shimara <@ raks>
rewrite: ruleset  9   input: shimara <@ raks>
rewrite: ruleset  9 returns: shimara <@ raks>
rewrite: ruleset  0 returns: $# ether $# mailhost $: shimara < @ raks >
>
```

See Chapter 3, “Customizing sendmail Configuration Files,” for a complete description of the diagnostic information.

▼ How to Verify Connections to Other Systems

To verify connections to other systems, you can use the `mconnect` program to open connections to other `sendmail` systems over the network. The `mconnect` program runs interactively. You can issue various diagnostic commands. See the `mconnect(1)` man page for a complete description. The example below verifies that mail to the user name `shamira` is deliverable.

```
$ mconnect raks
connecting to host raks (129.144.52.96), port 25
connection open
220 raks.Eng.Sun.COM Sendmail SMI-8.6/SMI-SVR4 ready at Tue, 25 Jul 1995 10:45:28 -0700
vrfy shamira
250 Michael Wong <shamira@raks.Eng.Sun.COM>
>
```

If you cannot use `mconnect` to connect to an `SMTP` port, check these conditions:

- Is the system load too high?
- Is the `sendmail` daemon running?
- Does the system have the appropriate `/etc/mail/sendmail.cf` file?
- Is port 25 (the port that `sendmail` uses) active?

Other Diagnostic Information

For other diagnostic information, check the following sources:

- Look at the `received` lines in the header of the message. These lines trace the route the message took as it was relayed. Note that in the UUCP network many sites do not update these lines, and in the Internet the lines often get rearranged. To straighten them out, look at the date and time in each line. Do not forget to account for time–zone differences.
- Look at the messages from `MAILER-DAEMON`. These typically report delivery problems.
- Check the system log that records delivery problems for your group of systems. The `sendmail` program always records what it is doing in the system log. You may want to modify the `crontab` file to run a shell script nightly that searches the log for `SYSERR` messages and mails any that it finds to the postmaster.

- Use the `mailstats` program to test mail types and determine the number of messages coming in and going out.

Customizing `sendmail` *Configuration Files*

3

The `sendmail` program is a mail transport agent that uses a configuration file to provide aliasing and forwarding, automatic routing to network gateways, and flexible configuration. The SunOS 5.x operating system supplies standard configuration files that most sites can use. Chapter 2, “Setting Up and Administering Mail Services,” explains how to set up an electronic mail system using the standard files. This chapter explains how to customize `sendmail` configuration files if you need to tailor them to fit your site’s needs.

3

`sendmail` *Overview*

The `sendmail` program can use different types of communications protocols, like TCP/IP and UUCP. It also implements an SMTP server, message queueing, and mailing lists. Name interpretation is controlled by a pattern-matching system that can handle both domain-based naming and improvised conventions.

The `sendmail` program can accept domain-based naming, as well as arbitrary (older) name syntaxes—resolving ambiguities by using heuristics you specify. `sendmail` can also convert messages between disparate naming schemes. The domain technique separates the issue of physical versus logical naming. See *TCP/IP and Data Communications Administration Guide* for a complete description of Internet domain–naming conventions.

Certain special cases can be handled by improvised techniques, like providing network names that appear local to hosts on other networks.

`sendmail` *Features*

The `sendmail` program provides the following features:

- It supports UNIX System V mail, UNIX version 7 mail, and Internet mail.
- It is reliable. It is designed to correctly deliver every message. No message should ever be completely lost.
- It uses existing software for delivery whenever possible.
- It can be configured to handle complex environments, including multiple connections to a single network type (like with UUCP or Ethernet). `sendmail` considers the contents of a name as well as its syntax to determine which mailer to use.
- It uses configuration files to control mail configuration.
- Groups can maintain their own mailing lists. Individuals can specify their own forwarding without modifying the domain-wide alias file (typically located in the domain-wide aliases maintained by NIS or NIS+).
- Each user can specify a custom mailer to process incoming mail, which can provide functions like returning an “I am on vacation” message. See the `vacation(1)` man page for more information.

Figure 3-1 shows how `sendmail` interacts with the other programs in the mail system.

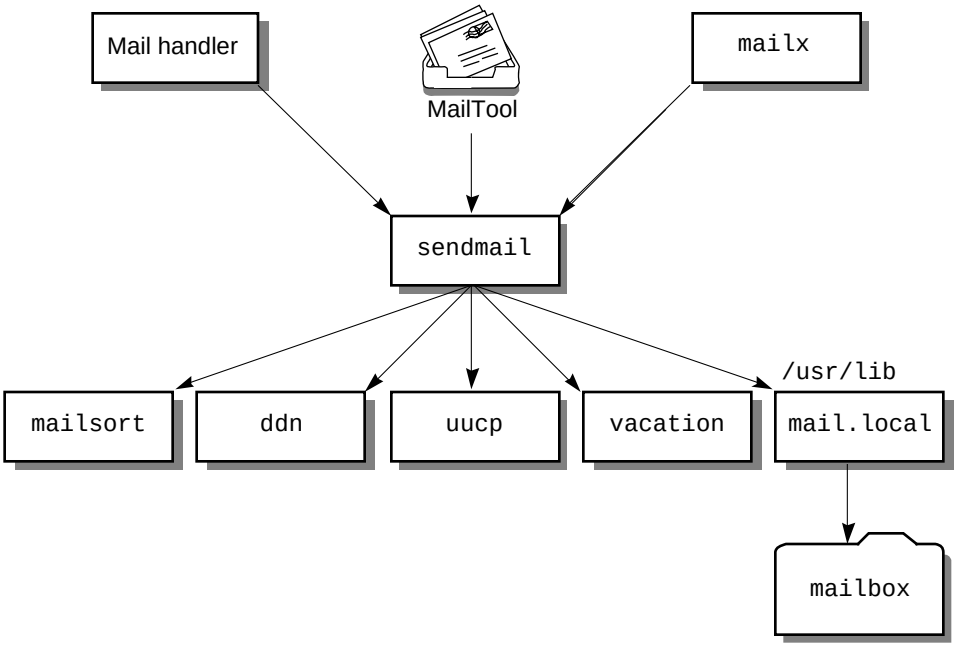


Figure 3-1 Interaction of `sendmail` With Other Mail Programs

The user interacts with a mail-generating and -sending program. When the mail is submitted, the mail-generating program calls `sendmail`, which routes the message to the correct mailers. Since some of the senders may be network servers and some of the mailers may be network clients, `sendmail` may be used as an Internet mail gateway.

`sendmail` *Functions*

The `sendmail` program is a message router that calls administrator-defined mailer programs to deliver messages. It collects a message from a program, like `mail`, edits the header of the message as required by the destination mailer,

and calls appropriate mailers to do delivery or queueing for network transmission. When mailing to a file, however, `sendmail` delivers directly. New mailers can be added at minimum cost.

`sendmail` *Interfaces*

The `sendmail` program communicates with the outside world in three ways:

- Using the conventional argument vector / exit status
- Using pairs of pipes
- Using SMTP over a TCP connection

Argument Vector/Exit Status

The standard way to communicate with a process is by using the argument vector (command name and arguments). The argument vector sends a list of recipients, and the message body is sent on the standard input. If problems occur, anything that the mailer prints is collected and returned to the sender. After the message is sent, the exit status from the mailer is collected and a diagnostic is printed, if appropriate.

SMTP Over Pipes

The SMTP protocol can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient names are passed to the mailer through the argument list. Instead, they are passed one at a time in commands sent to the processes' standard input. Anything appearing on the standard output must be a standard SMTP reply code.

SMTP Over a TCP Connection

This technique is similar to SMTP over pipes, except that it uses a TCP connection. SMTP over a TCP connection is normally used to connect to a `sendmail` process on another system. This method is exceptionally flexible because the mailer need not reside on the same machine.

How `sendmail` *Works*

The `sendmail` program collects a message from a program like `mailx` or `mailtool`, edits the message header as required by the destination mailer, and calls appropriate mailers to do delivery or queueing for network transmission.

Note – The `sendmail` program never edits or changes the body of a message. Any changes that it makes to interpret email addresses are made only in the header of the message.

Argument Processing and Address Parsing

When `sendmail` receives input, it collects recipient names (either from the command line or from the SMTP protocol) and generates two files. One is an *envelope* that contains the header and a list of recipients. The other file contains the body of the message. The `sendmail` program expands aliases, including mailing lists, and validates as much as possible of the remote recipient. Then `sendmail` checks syntax and verifies local recipients. Detailed checking of host names is deferred until delivery. As local recipients are verified, messages are forwarded to them.

After parsing the recipient lists, `sendmail` appends each name to both the envelope and the header of the message. When a name is aliased or forwarded, it retains the old name in the list and sets a flag to tell the delivery phase to ignore this recipient. The lists are kept free from duplicates, preventing alias loops and duplicate messages delivered to the same recipient, which can occur if a recipient is in two different alias groups.

Note – Users may receive duplicate copies of the same message when alias lists contain email addresses for the same person using different syntax. The `sendmail` program cannot always match the email addresses as duplicates of one another.

Message Collection

Once all recipient names are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body. The header and the body are separated by a blank line.

The header is formatted as a series of lines in this form:

```
field-name: field-value
```

For example, a sample header might be:

```
From: John Smith <Smith@colorado.edu>
```

field-value can be split across lines by starting the subsequent lines with a space or a tab. Some header fields have special internal meaning and have appropriate special processing. Other headers are simply passed through. Some header fields, like time stamps, may be added automatically.

No formatting requirements are imposed on the message body except that they must be lines of text. `sendmail` stores the header in memory and stores the body of the message in a temporary file. To simplify the program interface, the message is collected, even if no names are valid. If none of the names are valid, the message is returned to the sender with an error.

Note – With DeskSet™ Mail Tool, users can transmit binary data. However, it must be encoded by Mail Tool. The `sendmail` program does not automatically encode binary data. Refer to the `mailtool(1)` man page for information about how to encode and decode binary data messages.

The message body is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

Message Delivery

The send queue is grouped by the receiving host before transmission to implement message batching. An argument list is built as the scan proceeds. Mail being sent to files is detected during the scan of the send list..

After a connection is established, `sendmail` makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery is complete.

The sendmail program sends the message to the mailer using one of the same interfaces used to submit a message to sendmail (using the conventional UNIX argument vector/return status, communicating over a pair of UNIX pipes, or using SMTP over a TCP connection). Each copy of the message has a customized header.

Error Handling

When mail can't be delivered, the mailer catches and checks the status code, and a suitable error message is given as appropriate. The exit code must conform to a system standard. If a nonstandard exit code is used, sendmail transmits the message, "Services unavailable."

Queueing for Retransmission

If the mailer returns a "temporary failure" exit status, the message is queued. A control file is used to describe the recipients and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other parameter of the message. The header of the message is stored in the control file so that the associated data file in the queue is just the temporary file that was originally collected.

Return to Sender

If errors occur during processing, sendmail returns the message to the sender for retransmission. The letter may be mailed back or written to the dead.letter file in the sender's home directory.

Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a From: line and a Full-name: line can be merged under certain circumstances.

Configuration File

Almost all configuration information is read at runtime from a text file that includes macro definitions (the value of macros used internally), header declarations (the format of header lines that are specially processed and lines that are added or reformatted), mailer definitions (giving information like the location and characteristics of each mailer), and name-rewriting rules (a limited pattern-matching system used to rewrite names).

How `sendmail` *Is Implemented*

You can follow flag arguments with recipient name arguments unless you run `sendmail` in SMTP mode. In brief, the format of recipient names is:

- Anything in parentheses is thrown away (as a comment).
- Anything in angle brackets (<>) is preferred over anything else. This rule implements the Internet standard that sends names of the following form to the electronic *system-name* rather than the human *username*:

```
username <system-name>
```

- Double quotation marks (") quote phrases; backslashes (\) quote characters. Backslashes cause otherwise equivalent phrases to compare differently—for example, `user` and `"user"` are equivalent, but `\user` is different from either of them.

Parentheses, angle brackets, and double quotation marks must be properly balanced and nested. The rewriting rules control the rest of the needed processing.

Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to use `mailsort` to sort mail or to have the `vacation` program respond with an informational message when users are away.

Any name passing through the initial parsing algorithm as a local name is scanned for two special cases:

- If the prefix is a vertical bar (|), the rest of the name is processed as a shell command.
- If the user name begins with a slash (/), the name is used as a file name instead of a login name.

Configuration Overview

Configuration is controlled primarily by a configuration file read at startup. Adding mailers or changing the name-rewriting or routing information does not require recompiling. The configuration file encodes macro definitions, header definitions, mailer definitions, name-rewriting rules, and options.

Macros

Macros can be used in several ways. Certain macros transmit unstructured textual information into the mail system, like the name that `sendmail` will use to identify itself in error messages. Other macros are unused internally and can be used as shorthand in the configuration file.

Header Declarations

Header declarations inform `sendmail` of the format of known header lines. Knowledge of a few header lines is built into `sendmail`, like the `From:` and `Date:` lines.

Most configured headers are automatically inserted into the outgoing message if they do not exist in the incoming message. Certain headers are suppressed by some mailers.

Mailer Declarations

Mailer declarations specify the internal name of the mailer, some flags associated with the mailer, and an argument vector to be used on the call. This vector is expanded by macro before use.

Name-Rewriting Rules

Name-rewriting rules are the heart of name parsing in `sendmail`. They are an ordered list of pattern-replacement rules that are applied to each name. For example, rule set 0 determines which mailer to use. `sendmail` rewrites the message until it is in a form that can be parsed. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of names into different formats. For example, a name in this form

```
ucsfcgl!tef
```

might be mapped into

```
tef@ucsfcgl.UUCP
```

to conform to the internal syntax. Translations can also be done in the other direction for particular mailers.

Option Setting

Several options can be set from the configuration file. These include the path names of various support files, timeouts, default modes, and so forth.

Arguments to `sendmail`

Arguments to `sendmail` are listed and described in detail in the section “Command-Line Arguments to sendmail” on page 102. Some of the important arguments are described in this section.

Queue Interval

The `-q` flag defines how often `sendmail` runs the queue. If you run in mode `i` or `b` (the default), this time can be relatively long, because it is only relevant when a host that was down comes back up. If, however, you run in `q` mode,

the time should be relatively short, since the flag defines the maximum amount of time that a message may sit in the queue. Typical queue time is set to between 15 minutes (`-q15m`) and one hour (`-q1h`).

Daemon Mode

If you allow incoming mail over a TCP connection, you should have a daemon running. Set the `-bd` flag in your `/etc/rc3.d/S88sendmail` file.

You can combine the `-bd` flag and the `-q` flag in one call:

```
# /usr/lib/sendmail -bd -q30m
```

Debugging

`sendmail` has many debug flags, which you set using the `-d` option. Each debug flag has a number and a level, where higher levels mean "print more information." The convention is that you do not need to set levels greater than 9 unless you are debugging that particular piece of code. The syntax for debug flags is:

```
debug-flag  :                -d debug-list
debug-list  :          debug-option[,debug-option ] . . .
debug-option  :        debug-range[.debug-level ]
debug-range  :         integer |integer–integer
debug-level  :         integer
```

For example:

Table 3-1 Sample `sendmail` Debug Flags

Debugging Flag	Description
`-d12`	Set flag 12 to level 1
`-d12.3`	Set flag 12 to level 3
`-d3-17`	Set flags 3 through 17 to level 1
`-d3-17.4`	Set flags 3 through 17 to level 4

If you have source code, you can refer to the list of debug flags in the code.

Trying a Different Configuration File

You can specify an alternative configuration file by using the `-C` flag; for example, the command

```
# /usr/lib/sendmail -Ctest.cf
```

uses the configuration file `test.cf` instead of the default `/etc/mail/sendmail.cf`. If you do not define a value for the `-C` flag, it uses the `sendmail.cf` file in the current directory.

Tuning Configuration Parameters

You can tune several configuration parameters, depending on the requirements of your site. You can set most of these parameters by using an option in the configuration file. For example, the line `OT3d` sets option `T` to the value `3d` (three days).

Time Values

All time intervals use a syntax of numbers and letters. For example, `10m` is 10 minutes, and `2h30m` is two-and-one-half hours.

Table 3-2 lists the time symbols.

Table 3-2 Time Syntax Options

Code	Description
`s`	Seconds
`m`	Minutes
`h`	Hours
`d`	Days
`w`	Weeks

Queue Interval

The argument to the `-q` flag specifies how often `sendmail` runs the queue. It is usually set to between 15 minutes (`-q15m`) and one hour (`-q1h`).

Read Timeouts

The `Or` option in the configuration file sets the read timeout. The default read timeout is `Or15m`. Although it is technically unacceptable within the published protocols, `sendmail` may time out when reading the standard input or when reading from a remote SMTP server. If your site has problems with read timeouts, set the read timeout to a larger value, like one hour (`Or1h`), to reduce the chance of several idle daemons piling up on your system.

Message Timeouts

The `OT` option in the configuration file sets the message timeout. The default message timeout is three days (`OT3d`). After a message has been in the queue for the message timeout period, the sender is notified that the message could not be delivered.

You can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example, the command

```
# /usr/lib/sendmail -oT1d -q
```

runs the queue and flushes anything that is one day old or older.

Delivery Mode

The `Od` option in the configuration file sets the delivery mode. The default delivery mode is `Odbackground`. Delivery modes specify how quickly mail is delivered. Legal modes are shown in Table 3-3.

Table 3-3 Delivery–Mode Options

Code	Description
`i`	Delivers interactively (synchronously)
`b`	Delivers in background (asynchronously)
`q`	Queues only (do not deliver)

There are trade-offs. The interactive mode (`i`) passes the maximum amount of information to the sender, but it is hardly ever necessary. The queue mode (`q`) puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. The background mode (`b`), the default, is a good compromise.

Load Limiting

Central mail machines often can be overloaded. The best solution is to dedicate a more powerful machine to handling mail, but the load almost always expands to consume whatever resources are allocated.

The `sendmail` program enables you to limit the load. The goal of load limiting is to prevent wasting time during loaded periods by attempting to deliver large messages, messages to many recipients, and messages to sites that have been down for a long time.

Use the `Ox` and `OX` options to limit the load caused by `sendmail`. The default is not to set any load limits if no options are used. Both of these configuration options take an argument that is an integer load average. For example, if you specify `Ox4` and `OX8`, then the `x` load limiting will be used when the load is above four, and the `X` load limiting will be used when the load is above eight. When the load is above the value specified in the `X` option, the SMTP server does not accept connections from the network (locally originated mail and other mail like UUCP are not affected). The `x` option has a more subtle effect,

controlling whether messages are queued for later delivery or delivered immediately. The general idea is always to deliver “small” messages immediately and to defer “large” messages for delivery during off-peak periods.

The `Oq` option specifies the maximum size of a message that is delivered immediately. The “size” of the message includes not only the number of bytes in the message but also the penalty for a large number of recipients and for unsuccessful delivery attempts. The penalty per recipient is option value `y`, by default set to 1000. The penalty per delivery attempt is the option value `z`, by default set to 9000. The size limit also depends on the current load, so that more and more messages are queued as the load goes higher. If the load is one above the `x` threshold, then the limit is halved; if the load is two above the threshold, the limit is divided by three, and so on. Note that this limit also applies to messages being delivered when processing the queue.

Log Level

You can adjust the level of logging for `sendmail`. The levels are shown in Table 3-4.

Table 3-4 Log–Level Codes

Code	Description
`0`	No logging
`1`	Major problems only; serious system failures and potential security problems
`2`	Lost communications (network problems) and protocol failures
`3`	Other serious failures
`4`	Minor failures
`5`	Message collection statistics
`6`	Creating of error messages, VRFY and EXPN commands
`7`	Delivery failures (host or user unknown, and so forth.)
`8`	Successful deliveries
`9`	(Default) Messages being deferred (due to a host being down, and so forth.)
`10`	Database expansion (alias, forward, and userdb lookups)

Table 3-4 Log–Level Codes *(Continued)*

Code	Description
15	Automatic alias database rebuilds
20	Logs attempts to run locked queue files
30	Lost locks

Refer to the system error logging section in *System Administration Guide, Volume I* for more information.

File Modes

Certain files may have protections modes that control access. This section describes the modes that you can control from the `sendmail.cf` file. The modes you use depend on what functionality you want and the level of security you require.

`setuid`

By default, the `sendmail` program is executed with the user ID set to 0 (`setuid` to root) so that it can deliver to programs that might write in a user's home directory. When `sendmail` is ready to execute a mailer program, `sendmail` checks to see if the user ID is 0; if so, it resets the user ID and group ID to the values set by the `u` and `g` options in the configuration file. The user ID and group ID are both set to 1 (daemon). You can override these values by setting the `S` flag to the mailer (for mailers that are trusted, and must be called as root). However, mail processing is accounted to root rather than to the user sending the mail.

Temporary File Modes

The `OF` option sets the mode of all the temporary files that `sendmail` uses. The default value, `0600`, is appropriate for secure mail, and `0644` is more permissive. If you use the more permissive mode, it is not necessary to run `sendmail` as root at all (even when running the queue). Users will be able to read mail in the queue.

Should My Alias Database Be Writable?

One approach is to provide the alias database (`/etc/mail/aliases`) with mode `666`. If you use this approach, users can modify any list. However, you may want to limit the aliases that a user can change by putting them into a file that the user can edit and referencing this file from `/etc/mail/aliases`. Such references have the following format:

alias-name`::include:/`*filename*

`sendmail` *Configuration File*

This section describes the configuration file in detail, including hints for writing your own file.

The syntax of the configuration file is reasonably easy to parse, since parsing can be done every time `sendmail` starts. Unfortunately, this syntax sacrifices readability.

`sendmail` uses single letters for several different functions:

- Command-line flags
- Configuration options
- Queue–file line types
- Configuration–file line types
- Mailer field names
- Mailer flags
- Macro names
- Class names

The following sections provide an overview of the configuration file and details of its semantics. A copy of the default main.cf file is included in Appendix A, “sendmail Configuration File”.

Purpose of the `sendmail` *Configuration File*

The `sendmail` configuration file has three parts:

- Definitions of symbols, classes, options, and parameters
- Definitions of mailers and delivery programs
- Rule sets that determine the rules for rewriting addresses

You define symbols, classes, options, and parameters to set up the environment for `sendmail`, setting the options and defining a few critical macros.

You define your mailers and delivery programs so that `sendmail` knows which protocols to use and which delivery programs to interact with.

You define rewriting rules, grouped into rule sets, to transform addresses from one form to another. In general, each rule in a rule set is applied to a particular address. An address might be rewritten several times within a rule set.

There are seven standard rule sets, applied in the order shown in Table 3-5.

Table 3-5 Order of Application of Rule Sets

Rule set	Description
`3`	The first rule set applied. It tries to put the address into the form: *local-address@host-domain* .
`0`	Determines the destination and which mailer program to use to send the message. It resolves the destination into a triple (*mailer*, *host*, *user*).
`1`	Rewrites the sender address.
`S`	Specifies additional rule sets that enable the sender addresses to do final mailer-specific cleanup. These rule sets have different names for each mailer. `S`, for example, stands for a generic “sender.”
`2`	Rewrites the recipient address.
`R`	Specifies additional rule sets that enable the recipient addresses to do final mailer-specific cleanup. These rule sets have different names for each mailer. `R`, in this example, stands for a generic “recipient.”
`4`	Rewrites all addresses for the last time, usually from internal to external form.

Rule set 0 must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

Rewriting names in the message is typically done in two phases. The first phase uses rule set 3 to map names in any format into a *local-address@host-domain* form. The second phase converts map names to the canonical form into the syntax appropriate for the receiving mailer. `sendmail` rewrites names in

three subphases. Rule sets 1 and 2 are applied to all sender and recipient names, respectively. Mailer-specific rule sets are specified during mailer definition. Finally, rule set 4 is applied to do any conversion to external form.

RFC 822 describes the format of the mail message itself. `sendmail` follows this RFC closely, to the extent that many of the standards described in this document cannot be changed without changing the code. In particular, the following characters have special interpretations:

```
< > ( ) " \
```

Caution – Use the RFC 822 special characters `< > ( ) " \` only for their designated purposes. Information between parentheses, `()`, is reserved for comments or personal names. Information between angle brackets, `<>`, is reserved for *local-address@host-domain* addresses.

`sendmail` *Configuration File Syntax*

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a pound sign (#) are ignored.

`D` *and* `L` *— Define Macro*

Macros are named with a single character. Although a macro can be defined with any character from the complete ASCII set, use only uppercase letters for macros that you define. However, do not use characters like `M`, `R`, `L`, `G` and `V` that are already used by `sendmail`. Lowercase letters and special symbols are used internally.

There are two ways you can define macros:

- `D` assigns the value directly specified.
- `L` assigns the value looked up in the `sendmailvars` database (either the NIS+ sendmailvars table or `/etc/mail/sendmailvars` file). The `L` command is classified as a Sun uncommitted interface.

The syntax for `D` macro definitions is:

```
DXval
```

where *X* is a letter defining the macro and *val* is the value it should have. No spaces are allowed. Macros can be inserted in most places using the escape sequence `$`*X*.

Here are examples of `D` macro definitions from the configuration file:

```
DRmailhost
Dmeng.acme.com
```

The variable `R` is set to contain the value `mailhost` and the internal variable `m` is set to contain the value `eng.acme.com`.

The `m` macro defines the mail domain. If it is not defined, the name service domain name is used with the first component stripped off. For example, `ecd.east.acme.com` becomes `east.acme.com`. An even more flexible way to define the mail domain name is to use an `L` macro definition, as shown below.

The syntax for an `L` macro definition is:

```
LXsearch-key
```

where *X* is the name of the macro and *search-key* is looked up in the `sendmailvars` database. The value found in the entry located by the search key is assigned to *X*.

Here is an example of an internal `L` macro definition from the configuration file:

```
Lmmaildomain
```

The variable m is set to the value found in the sendmailvars database using maildomain as the search key. If the entry in the sendmailvars database appears as follows:

```
maildomain        eng.acme.com
```

The value of m becomes eng.acme.com.

Note – sendmail uses the sendmailvars entry in the /etc/nsswitch.conf file to determine the order in which to search the name space and /etc/mail/sendmailvars file.

C, F, *and* G—*Define Classes*

You can define classes of words to match the left-hand side of address–rewriting rules. For example, you might create a class of all local names for this site so that you can eliminate attempts to send mail to yourself.

You can give classes names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

There are three ways to define classes:

- C assigns the values directly specified.
- F reads in the values from another file or from another command.
- G assigns the values looked up in the sendmailvars database (either the NIS+ sendmailvars table or /etc/mail/sendmailvars file). The G command is classified as a Sun uncommitted interface.

The syntaxes of the different forms of class definition are:

C*X* *word1* *word2*

F*X* *file* [*pattern]*

F*X* | *command*

G*Xsearch-key*

The first form defines the class *X* to match any of the named words. The second form reads words from the file into the class *X*, for example, FC /.rhosts. The *pattern* argument to the F*C* class is used with scanf to read

from the file; otherwise, the first word from each line is used. The third form executes the given command and reads the elements of the class from standard output of the command. For example:

```
FC | awk '{print $2}' /etc/hosts
```

The fourth form reads the elements of the class from the entry in the `sendmailvars` database pointed to by the search key. For example,

```
GVuucp-list
```

gets the definition of class `V` from the `uucp-list` entry in the `sendmailvars` database.

If the entry in the `sendmailvars` database appears as follows:

```
uucp-list        sunmoon hugo comic
```

the value of `V` becomes `sunmoon hugo comic`.

Note – `sendmail` uses the `sendmailvars` entry in the `/etc/nsswitch.conf` file to determine the order in which to search the name space and `/etc/mail/sendmailvars` file.

You could split class definitions among multiple lines. For example, the following:

```
CHmonet ucbmonet
```

is equivalent to:

```
CHmonet
CHucbmonet
```

O— *Set Option*

You can set several options (not to be confused with mailer flags or command-line arguments) from a configuration file. Options are also represented by single characters. The syntax of this line is:

```
Oc value
```

This sets option *c* to *value*. Depending on the option, *value* may be a string, an integer, a Boolean (with legal values `t`, `T`, `f`, or `F`—the default is "true''), or a time interval. See "Configuration Options to sendmail" on page 103 for the list of options.

P— *Precedence Definitions*

You can define values for the `Precedence:` field using the `P` control line. The syntax of this field is:

```
Pname=num
```

When the *name* is found in a `Precedence:` field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages are not returned. The default precedence is 0 (zero). For example:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

T— *Define Trusted Users*

This configuration line has been deleted. It will be accepted but will be ignored.

H— *Define Header*

The format of the header lines is defined by the `H` line. The syntax of this line is:

```
H[c ?c mflagsc ?]c hnamec :c htemplate
```

Continuation lines in this specification are inserted directly into the outgoing message. The *htemplate* is macro-expanded before it is inserted into the message. If the expansion is empty, the header line is not included. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition before this header can be automatically output. If one of these headers is in the input, it is directed to the output regardless of these flags.

Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into `sendmail` that cannot be changed without changing the code. These built-in features are described in the following list

- `Return-Receipt-To:` – If this header is sent, a message will be sent to any specified names when the delivery is complete. The mailer must have the `l` flag (local delivery) set in the mailer descriptor.
- `Errors-To:` – If errors occur anywhere during processing, this header sends error messages to the listed names rather than to the sender. Use this header line for mailing lists.
- `To:` – If a message comes in with no recipients listed in the message (in a `To:`, `Cc:`, or `Bcc:` line) then `sendmail` adds an `Apparently To:` header line for each recipient specified on the `sendmail` command line.

R *and* S—*Rewriting Rules*

Address parsing is done according to the rewriting rules, which is a simple pattern-matching and replacement system. `sendmail` scans through the set of rewriting rules looking for a match on the left-hand side (LHS) of the rule. When a rule matches, the name is replaced by the right-hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics and may be referenced by the mailer definitions or by other rewriting sets.

For example:

```
Sn
```

sets the current rule set being collected to *n*. If you begin a rule set more than once, the new definition overwrites the old definition.

R is used to define a rule in the rule set. The syntax of the R line is:

```
Rlhs      rhs   comments
```

The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

Here is an example of how a rule definition might look:

```
# handle "from:<>" special case
R<>        $@@            turn into magic token
```

The fields must be separated by at least one tab character; you may use embedded spaces in the fields.

M — *Define Mailer*

Programs and interfaces to mailers are defined on this line. The format is:

```
Mc name, c
{c  field=value}*
```

where *name* is the name of the mailer (used in error messages) and the *field=value* pairs define attributes of the mailer. The fields are shown in Table 3-6.

Table 3-6 Mailer Definition Fields

Field	Description
A	An argument vector to pass to this mailer
D	The working directory for the mailer
E	The end-of-line string for this mailer
F	Special flags for this mailer
L	The maximum line length in the message body
M	The maximum message length to this mailer
P	The path name of the mailer
R	A rewriting rule set for recipient names
S	A rewriting rule set for sender names

Address–Rewriting Rules

This section describes the details of rewriting rules and mailer descriptions.

Special Macros, Conditionals

Special macros are referenced with the construct $*x*, where *x* is the name of the macro to be matched (LHS) or inserted (RHS). Lowercase letters are reserved for special semantics, and some special characters are reserved to provide conditionals.

The macros shown in Table 3-7 *must* be defined to transmit information into `sendmail`.

Table 3-7 Required `sendmail` Macros

Macro	Description
`$a`	Origination date in ARPANET format
`$b`	Current date in ARPANET format
`$c`	Hop count
`$d`	Date in UNIX (`ctime`) format
`$e`	Printed out when SMTP starts
`$f`	Sender from name
`$g`	Sender name relative to the recipient
`$h`	Recipient host
`$i`	Queue ID
`$j`	The official domain name for this site; should be the first word of the `$e` macro; `$j` should be in domain name format
`$k`	The UUCP node name (from `uname`)
`$l`	The format of the UNIX `From` line
`$m`	The domain part of the `gethostname` return value.
`$n`	The name of the daemon (for error messages)
`$o`	List of characters that are considered tokens
`$p`	`sendmail`'s process ID.
`$q`	Default format of the sender address. Specifies how a sender should appear in a message when it is created
`$r`	Protocol used
`$s`	Sender's host name
`$t`	Numeric representation of the current time
`$u`	Recipient user
`$v`	Version number of `sendmail`
`$w`	Host name of this site

Table 3-7 Required `sendmail` Macros *(Continued)*

Macro	Description
`$x`	Full name of the sender
`$z`	Home directory of the recipient
`$-`	Sender address

For example:

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g $d
Do.:%@!^=/
Dq$g$?x ($x)$.
Dj$H.$D
```

You should not need to change any of these macros except under unusual circumstances. For example, you might want to change the first line which defines the banner for security. You might want to change the last two lines to make several hosts look like one host.

An acceptable alternative for the `$q` macro is:

```
$?x$x $.<$g>
```

This entry corresponds to the following two formats:

```
doe@acme.com (John Doe)
John Doe <doe@acme.com>
```

Some macros are defined by `sendmail` for use in mailer arguments or for other contexts. These macros are shown in Table 3-8.

Table 3-8 Additional `sendmail` Macro Definitions

Macro	Description
`m`	Domain name
`p`	`sendmail` process ID

You can use three types of dates. The $a and $b macros are in ARPANET format; $a is the time as extracted from the Date: line of the message (if there was one), and $b is the current date and time (used for postmarks). If no Date: line is found in the incoming message, $a is also set to the current time. The $d macro is equivalent to the $a macro in UNIX (ctime) format.

The $f macro is the ID of the sender as *originally determined*; for a message mailed to a specific host, the $g macro is set to the name of the sender *relative to the recipient*. For example, suppose the sender eric sends a message to bollard@matisse from the machine ucbarpa. The value of $f will be eric and the value of $g will be eric@ucbarpa.

The $x macro is set to the full name of the sender, which can be determined in several ways. It can be passed as a flag to sendmail (from the value of the Full-name: line in the header or use the comment field of a From: line). If the name can't be determined from the Full-name or From lines, and if the message is being originated locally, the full name is looked up in the /etc/passwd file. It can also be read from the name environment variable.

When a message is sent, the $h, $u, and $z macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the $@ and $: part of the rewriting rules, respectively.

The $p and $t macros are used to create unique strings (for example, for the Message-Id: field). The $i macro is set to the queue ID on this host; if put into the time stamp line, it can be useful for tracking messages. The $v macro is set to be the version number of sendmail; this is normally put in time stamps and is extremely useful for debugging. It can, however, be a security risk. The $w macro is set to the primary name of this host, as given by gethostname() and gethostbyname(). The $c field is set to the "hop count"; that is, the number of times this message has been processed, which can be determined by counting the time stamps in the message.

The $r and $s fields are set to the protocol used to communicate with sendmail and the sending host name.

You can specify conditionals by using the syntax:

```
$?x text1 $| text2 $
```

This inserts *text1* if the macro $x is set, and *text2* otherwise. The else (c $|) clause may be omitted.

Special Classes

The class `$=w` is the set of all names by which this host is known. It can be used to delete local host names. The class `$=m` is set to the domain name.

Left-Hand Side of Address–Rewriting Rules

The left-hand side of rewriting rules contains a pattern. Normal words are matched directly. Dollar signs introduce "metasymbols," which match things other than simple words, like macros or classes. The metasymbols are shown in Table 3-9.

Table 3-9 `sendmail` Left-Hand Side Metasymbols

Symbol	Matches
`$*`	Zero or more tokens
`$+`	One or more tokens
`$–`	Exactly one token
`$=`*x*	Any string in class *x*
`$~`*x*	Any token not in class *x*
`$`*x*	Macro *x*

If any of the patterns matches, it is assigned to the symbol `$`*n* for replacement on the right-hand side, where *n* is the index in the LHS. For example, the LHS

```
$–:$+
```

rules can be applied to this input:

```
JUPITER:eric
```

The rule will match, and the values passed to the RHS will be:

```
$1  JUPITER
$2  eric
```

Solaris Specific Rules

Several special rules have been added to work with the name space. These rules are not generic to sendmail, so can only be used on systems running the Solaris software.

The special form $%y matches any host name in the hosts information in the name space. Either local or remote hosts can be matched using this rule. It does a most-to-least multi-token match, so it will handle fully qualified hostnames as well as a short local hostname.

The $%x form will match MX records through DNS. This will succeed even if the A-record does not exist in the DNS database.

The $%l matches any fully qualified host in the local domain. If NIS or local files are being used, this means that the hostname in the name space must include the local domain name or DNS forwarding has to be turned on. The NIS+ name space will qualify the hostname appropriately, without any changes.

To use $%l in an NIS environment in which DNS forwarding can not be set up and the name space can not be changed to use fully qualified hostnames, add the following line into the configuration file:

```
DAhosts.byname
```

and replace all occurrences of %l with %*A*. Any non-conflicting letter can be used in place of the *A*. This will turn off the need to look up fully qualified names as long as the target host can be resolved to an IP address and it is a single token name. All resolved addresses will be assumed to be local, so make sure that the name space does not contain any single token host entries that are external to the mail domain.

Right-Hand Side of Address Rewriting Rules

When the left-hand side of a rewriting rule matches, the input is replaced by the right-hand side. Tokens are copied directly from the right-hand side, unless they begin with a dollar sign, in which case they are treated as macros and expanded.

Metasymbols for more complicated substitutions are shown in Table 3-10.

Table 3-10 `sendmail` Right-Hand Side Metasymbols

Symbol	Description
$*x*	Expands macro *x*
$*n*	Substitutes indefinite token *n* from LHS; *n* is a digit
$>*n*	Calls rule set *n; n* is a digit
$#*mailer*	Resolves to *mailer*
$@*host*	Specifies *host*
$:*user*	Specifies *user*
$[*host*$]	Maps to primary host name
${*x name*$}	Maps name through NIS map or NIS+ table $*x*; if the map name begins with `rev, sendmail` will reverse the aliases

The $*n* (*n* being a digit) syntax substitutes the corresponding value from a $+, $–, $*, or $=® match on the LHS. It may be used anywhere.

The $> *n* syntax substitutes the remainder of the line as usual and then passes it to rule set *n*. The final value of rule set *n* then becomes the substitution for this rule (like a procedure or function call).

Only use the $# syntax in rule set 0. Evaluation of the rule set stops immediately, and signals are sent to `sendmail` that the name has completely resolved. The complete syntax is:

```
$#mailer$@host$:user
```

This specifies the {`mailer`, `host`, `user`} triple necessary to direct the mailer. More processing may then take place, depending on the mailer. For example, local names are aliased.

A right-hand side may also be preceded by a $@ or a $: to control evaluation. A $@ prefix returns the remainder of the right-hand side as the value. A $: prefix terminates the rule immediately, but the rule set continues. Thus it can be used to limit a rule to one application. Neither prefix affects the result of the right-hand side expansion.

The $@ and $: prefixes can precede a $> spec. For example:

```
R$+   $:$>7$1
```

matches anything, passes that to rule set 7, and continues; the $: is necessary to avoid an infinite loop. The $[*host*]$ syntax replaces the host name with the "official" or primary host name, the one listed first in the hosts.byname NIS map, or /etc/hosts if not running NIS. It is used to eliminate nicknames for hosts. The ${x *name* $} syntax replaces the string by the result of the nis_map_name indicated in macro $x.

The rule below can be added to ruleset 0 to forward mail to the mailhost from a system (like a router) which man not be able to route mail itself:

```
R$*<@$*.$m>    $#ether $@mailhost $:$1<@$2.$m>
```

Semantics of Rewriting Rule Sets

Five rewriting sets have specific semantics, as shown in Figure 3-2 on page 92.

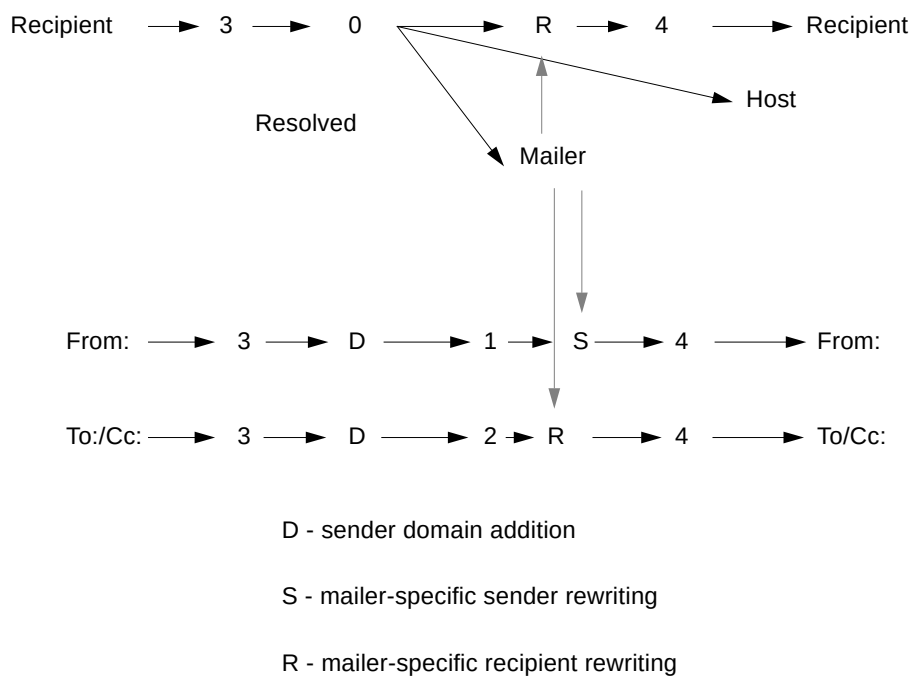


Figure 3-2 Rewriting Set Semantics

Rule set 3 is applied by `sendmail` before `sendmail` does anything with any name. Rule set 3 should turn the name into a form, with the basic syntax:

local-part@*host-domain-spec*

If no @ sign is specified, then the *host-domain-spec* may be appended from the sender name (if the `C` flag is set in the mailer definition corresponding to the *sending* mailer).

Rule set 0 is applied after rule set 3 to names that are actually going to specify recipients. It must resolve to a *mailer*, *host*, *user* triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the `$h` macro for use in the argument expansion of the specified mailer; the *user* is defined into `$u`.

Rule set 1 is applied to all `From:` recipient names, and rule set 2 is applied to all `To:` and `Cc:` lines. Then the rule sets specified in the mailer definition line (and `R=`) are applied. This process is done many times for one message, depending on how many mailers the message is routed to by rule set 0.

Rule set 4 is applied last to all names in the message. It is typically used to translate internal to external form.

`error` *Mailer*

You can use the mailer with the special name `error` in rule set 0 to generate a user error message. The user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule generates the specified error, if the LHS matches.

Semantics of Mailer Descriptions

Each mailer has an internal name. It can be arbitrary, except that the names `local` and `prog` must be defined first and second, respectively. Rule set 0 resolves names to this mailer name (and a host and user name).

Give the path name of the mailer in the `P` field. If this mailer will be accessed by way of a TCP connection, use the string `[TCP]` instead.

Define the mailer flags in the `F` field. Specify an `f` or `r` flag to pass the name of the sender as a `-f` or `-r` flag respectively. These flags are passed only if they were passed to `sendmail`, so that mailers that give errors under some circumstances can be placated. In some cases, you may be able to specify `-f$g` in the `argv` template. If the mailer must be called as root, and `sendmail` is running `setuid` to root, use the `S` flag; it will not reset the user ID before calling the mailer. If this mailer is local (that is, will perform final delivery rather than another network hop), use the flag. Quoted characters (backslashes and double quotation marks) can be stripped from names if the `s` flag is specified; if it is not specified, they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction, use the `m` flag. If this flag is on, then the `argv` template containing `$u` is repeated

for each unique user on a given host. The `e` flag marks the mailer as being "expensive," and `sendmail` defers connection until a queue run. Note that the `c` configuration option must also be set.

The `C` flag is a useful case. It applies to the mailer from which the message is sent, rather than the mailer where the message is received. If set, the domain specification of the sender (that is, the *@host.domain* part) is saved and appended to any names in the message that do not already contain a domain specification. For example, a message in this form:

```
From: eric@jupiter
To: joe@saturn, sam
```

is modified to

```
From: eric@jupiter
To: joe@saturn, sam@ganymede
```

if and only if the `C` flag is defined in the mailer corresponding to `eric@jupiter`.

The `S` and `R` fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient names, respectively. These are applied after the sending domain is appended and the general rewriting sets (number one or two) are applied, but before the output rewrite (rule set four) is applied. A typical use is to append the current domain to names that do not already have a domain. For example, a header in this form:

```
From: eric@host
```

might be changed to

```
From: eric@host.colorado.edu
```

or

```
From: saturn!eric
```

depending on the domain into which it is being shipped. These sets can also be used to do special-purpose output-rewriting in cooperation with rule set 4.

Table 3-11 includes additional flags that may be used in the configuration file.

Table 3-11 Additional Flags for the Mailer Description

Field Name	Used To
E	Defines the string to use as an end-of-line indication. A string containing `return` and `newline` is the default (if using TCP, otherwise just a `newline` indicates end-of-line. You can use the `print` backslash escapes (`/r`, `/n` `/f`, `/b`).
A	Specifies an argument vector template. It may have embedded spaces. The template is macro-expanded before being passed to the mailer. Useful macros include `$h`, the host name resolved by rule set 0, and `$u`, the user name (or names) resolved. If there is no argument with a `$u` macro in it, `sendmail` uses SMTP to communicate with the mailer. If the path name for this mailer is TCP, use the argument vector: `TCP $h [` *port* `],` where *port* is the optional port number to connect to.
L	Specifies the maximum length of the `$u` macro passed to the mailer. To make UUCP mail more efficient, the `L` field can be used with the `m` flag to send mail to multiple recipients with one call to the mailer, while avoiding mailer limitations on argument length. `$u` always expands to at least one recipient even if that recipient exceeds the `L=` limit.

For example, the specification:

```
Mlocal, P=/bin/mail, F=flsSDFMmnP, S=10, R=20, A=mail -d $u
Mprog,  P=/bin/sh,   F=lsDFMeuP,  S=10, R=20, A=sh -c $u
```

specifies a mailer for local delivery and a mailer for Ethernet delivery. The first is called `local`, is located in the file `/bin/mail`, takes a `-F` flag, does local delivery, strips quotes from names, and delivers mail to multiple users at once.

It applies rule set 10 to sender names in the message and applies rule set 20 to recipient names. The argument vector to send to a message is the word `mail`, the word `–d`, and words containing the name of the receiving user. If a `-r` flag is inserted, it is between the words `mail` and `–d`.

The second mailer is called `ether`.It is connected through TCP and can handle multiple users at once. It defers connections and appends any domain from the sender name to any receiver name without a domain; it processes sender names by rule set 11 and recipient names by rule set 21. Messages passed through this mailer have a 100,000-byte limit.

Building a Configuration File

Building a configuration file “from scratch” is a complex task. Fortunately, it is almost never necessary; you can accommodate almost every situation by changing an existing file. In any case, it is critical that you understand what it is that you are trying to do and come up with a policy statement for the delivery of mail. This section explains the purpose of a configuration file and gives you some ideas about policies.

Domains and Policies

RFC 1006 describes domain-based naming. RFC 822 touches on this issue as well. Essentially, each host is given a name that is a right-to-left dot-qualified pseudo-path from a distinguished root. The elements of the path are organizational entities, not physical networks.

RFC 822 and RFC 976 specify how certain sorts of addresses should be parsed. You can configure `sendmail` to follow or ignore these rules.

How to Proceed

After you have established a policy, examine the available configuration files to decide if you can use major parts of any of them. Even under the worst of conditions, there is a fair amount of boilerplate information that can be collected safely.

The next step is to build rule set 3, which specifies a rule set for your individual mailers. Building rule set 3 is the hardest part of the job. Here are some guidelines:

- Beware of doing too much to the name in this rule set, since anything you do will be reflected in the message.
- Do not strip local domains in this rule set. Doing so can leave you with names that have no domain specifications at all. `sendmail` appends the sending domain to names with no domain, which can change the semantics of names.
- Do not provide fully qualified domains in this rule set. Although technically correct, fully qualified domain names can lead to unnecessarily long names reflected into messages. The SunOS configuration files define rule set 9 to qualify domain names and strip local domains. Rule set 9 is called from rule set 0 to get all names into a cleaner form.

After you have rule set 3 finished, the other rule sets should be relatively simple. Examine the supplied configuration files for hints.

Testing the Rewriting Rules — the `-bt` *Flag*

When you build a configuration file, you can do a certain amount of testing using the test mode of `sendmail`. For example, you could invoke `sendmail` as:

```
% sendmail -bt -Ctest.cf
```

which would read the configuration file `test.cf` and enter test mode. For example:

```
ADDRESS TEST MODE
Enter <ruleset> <name>
>
```

In this mode, you enter lines in this form:

```
ADDRESS TEST MODE
Enter <ruleset> <name>
> rwset   name
```

where *rwset* is the rewriting set you want to use and *name* is a name to which to apply the set. Test mode shows you the steps it takes as it proceeds and shows you the final name. You may use a comma-separated list of *rwsets* for sequential application of rules to an input. For example:

```
ADDRESS TEST MODE
Enter <ruleset> <name>
> 3,1,21,4 jupiter:smith
```

First apply rule set 3 to the input `monet:bollard`. Rule set 1 is then applied to the output of rule set 3, followed similarly by rule sets 21 and 24.

If you need more detail, you can also use the `-d21` flag to turn on more debugging. For example, the command:

```
% sendmail -bt -d21.99
```

turns on large amount of information; a single–word name may result in several pages worth of information.

How `sendmail` *Interacts With a Name Service*

Mail domain is a concept used by the standard `sendmail.cf` file to determine whether mail should be delivered directly or through the mail host. Intra-domain mail is delivered through direct SMTP connection, while inter-domain mail is forwarded to a mail host.

In a secure network, only a few selected hosts are authorized to generate packets targeted to external destinations. Even if a host has the IP address of the remote host external to the mail domain, this does not guarantee that an SMTP connection can be established. The standard `sendmail.cf` assumes the following:

- The current host is not authorized to send packets directly to a host outside the mail domain.
- Mail host is capable of forwarding the mail to an authorized host that can transmit packets directly to an external host. (In fact, the mail host may itself be an authorized host.)

Given these assumptions, it is the responsibility of the mail host to deliver or forward inter-domain mail.

Setting Up `sendmail` *Requirements for Name Services*

`sendmail` imposes various requirements on name services. This section explains these requirements and how to satisfy them. For more information, refer to the `in.named(1M)`, `nis+(1)`, `nisaddent(1M)`, and `nsswitch.conf(4)` man pages.

The Mail Domain Name

The mail domain name must be a suffix of the name service domain. For example, if the domain name of the name service is `A.B.C.D`, then the mail domain name could be one of the following:

- `A.B.C.D`
- `B.C.D`
- `C.D`
- `D`

When first established, the mail domain name is identical to the name service domain. As the network grows larger, the name service domain is divided into more manageable pieces. However, the mail domain remains undivided to provide consistent aliasing.

If you are setting up NIS as the primary name service, `sendmail` automatically strips off the first component of the NIS domain name and uses the result as mail domain name. For example, `ebs.admin.acme.com` becomes `admin.acme.com`. No special command is needed.

If you are setting up NIS+ as your primary name service, a hierarchical namespace is required by NIS+. `sendmail` can look up the mail domain from the NIS+ sendmailvars table, a two–column NIS+ table with one key column and one value column. To set up your mail domain, you must add one entry to this table. This entry should have the key column set to the literal string

maildomain and the value column set to the your mail domain name (for example, admin.acme.com). Although NIS+ allows any string in the sendmailvars table, the suffix rule still applies for the mail system to work correctly. You can use nistbladm to add the maildomail entry to the sendmailvars table. For example, nistbladm -A \ key="maildomain" value=<mail domain> \ sendmailvars.org_dir.<NIS+ domain>. Note that this mail domain is a suffix of the NIS+ domain.

The Host Namespace Data

The host table or map in the name service must be set up to support three types of gethostbyname() queries:

- mailhost – Many name service configurations satisfy this requirement. Several typical configurations are described below.
  - *NIS with DNS forwarding* – When the DNS forwarding feature is turned on, queries that NIS cannot resolve are forwarded to DNS.
  - *NIS+ in conjunction with DNS* – If your network uses both NIS+ and DNS as the source for the host database, you can put the mailhost entry in either the NIS+ or DNS host table. Make sure that your users list NIS+ and DNS as the source for the host database in the /etc/nsswitch.conf file.
  - *Using NIS+ or NIS as the only name service* – If your network uses only one of these name services, you must have a mailhost entry in the NIS host map or the NIS+ hosts table.
- Full host name (for example, smith.admin.acme.com) – Many name service configurations satisfy this requirement:
  - *NIS with DNS forwarding* – Although NIS does not understand full host names, DNS does. This requirement is satisfied when you follow the regular procedure for setting up NIS and DNS.
  - *NIS+ and DNS* – Both NIS+ and DNS understand full host names. Following the regular NIS+ and DNS setup procedures satisfies this requirement.
  - *Using NIS+ as the only name service* – NIS+ understands the full host name. Following the regular NIS+ satisfies this requirement.
  - *Using NIS as the only name service* – This is a special case. The normal NIS setup does not understand the full host name. Rather than trying to make NIS understand the full host name, turn off this requirement from the sendmail side by editing the sendmail.cf file and replacing all occurrences of %l with %y. This turns off sendmail's inter-domain mail detection. As long as the target host can be resolved to a IP address, a

direct SMTP delivery will be attempted. Make sure that your NIS host map does not contain any host entry that is external to the current mail domain. Otherwise, you will need to further customize the `sendmail.cf` file.

- Short host name (for example, `smith`) – `sendmail` must connect to the mail host to forward external mail. To determine if a mail address is within the current mail domain, `gethostbyname()` is invoked with the full host name. If the entry is found, the address is considered internal.

  NIS, NIS+, and DNS all support `gethostbyname()` with a short host name as an argument, so this requirement is automatically satisfied.

`gethostbyname()` with full and short host name should yield consistent results. For example, `gethostbyname(smith.admin.acme.com)` should return the same result as `gethostbyname(smith)` as long as both functions are called from the mail domain `admin.acme.com`.

- *If you are using DNS in conjunction with NIS or NIS+*, for every host entry in the NIS or NIS+ host table, you must have a corresponding host entry in DNS.
- *If you are using NIS*, see above for a description of how to turn off `gethostbyname()` with a full host name.
- *If you are using NIS+*, see below.

For all name service domains under a common mail domain, `gethostbyname()` with a short host name should yield the same result. For example, given the mail domain `smith.admin.acme.com`, `gethostbyname(smith)` should return the same result calling from either domain `ebb.admin.acme.com` or `esg.admin.acme.com`.

The mail domain name is usually shorter than the name service domain, giving this requirement special implications for various name services:

- *NIS* – All NIS host maps under a common mail domain should have the same set of host entries. For example, the host map in the `ebs.admin.acme.com` domain should be the same as the host map in the `esg.admin.acme.com`. Otherwise, one address may work in one NIS domain but fail in the other NIS domain.
- *NIS+* – To satisfy this requirement, you may duplicate the host table as described in the NIS information above, or you can enter all host entries in the user name service domains into a master host table at mail domain level.

Because you are merging (logical or physically) multiple host tables into one host table, the same host name cannot be reused in the multiple name service domain sharing a common mail domain.

Command-Line Arguments to `sendmail`

Command-line arguments can be used when running `/usr/lib/sendmail` on a command line or can be entered into the mail startup script, `/etc/init.d/sendmail`. All of these arguments are listed in Table B-1 on page 117. Several of the most commonly used arguments are documented below.

The sendmail daemon is normally started with the following command:

```
/usr/lib/sendmail -bd -q1h
```

The `-b` option is used to select the operation mode that `sendmail` is running in. In this case, the daemon mode is selected. Other operational modes that can be selected are initialize the alias database (`-bi`), print the mail queue (`-bp`), or verify the recipients (`-bv`). The action of these arguments can be duplicated with `newaliases`, `mailq` and `mconnect`.

The second set of arguments indicates that any queued mail should be re-processed at one hour time intervals. This can be decreased to thirty minutes by using `-q30m`.

Using this argument, `-C`*file*, will allow testing of a different `sendmail` configuration file. Other options that are useful when testing or debugging `sendmail`, include `-d` which will select a debugging level and `-v` which selects the verbose mode.

The -q argument can also be used to limit the jobs that are processed. The following argument will select only those messages to a user named `mary`, `-qRmary`. This argument also allows limitation by sender, `-qS`*name*, or by message-id, `-qI`*number*.

The -o argument can be used to reset a configuration option. For instance, `-qQ`*/var/oldmail* will run `sendmail` using */var/oldmail* as the queue directory. The configuration options that can be changed with this argument are discussed in more depth in the next section.

Configuration Options to `sendmail`

All of the configuration options for `sendmail` are included in Table B-2 on page 119. These can be set using the `-o` flag when starting `sendmail` or by using the O line in the configuration file.

The A*file* option is used to change the location of the alias file, which is normally set to `/etc/mail/aliases`.

The hop count can be reset using the `h#` option. The hop count determines the maximum number of hosts a particular message will be routed through before it is sent back to the sender. If a mail message gets into an alias loop, the hop count determines the number of hosts the message will try to go to before the message fails. The default hop count is set to 30, which may be too high for very small domains or email domains that do not have many levels.

There are two options which can help control the system load. The `X#` option sets the maximum load average value and will cause the daemon to refuse any incoming SMTP connections if the load average is above the value designated by #. The default value is 12. This value can be increased if the mail host can take the load. Using `X0` will refuse all SMTP connections.

The `x#` option will queue the mail when the load average specified by # is reached instead of refusing it. Make sure that the server has enough disk space to hold all of the queued mail before adjusting this value.

Both of the options which work with the system load have a delay in response, so that restarting `sendmail` with new load average values, will not immediately remedy all load problems.

Mailer Flags

A complete list of the flags you can set in the mailer description are described in Table B-3 on page 123. These flags are set in the configuration file.

The `m` flag allows the mailer to batch a mail message to multiple users if they are all receiving mail on the same host. This can improve response time. It can cause problems if the mail can not reach one of the users. This will generate an error, which will cause for the mail to be resent, and can generate duplicate mail. Remove this flag from the configuration file if potential problems associated with duplicate mail are too great.

The `n` flag selects to not add a UNIX-style `From` line on the front of the message. This is already done by `/usr/lib/mail.local`, so it should not be necessary for the mailer to do this.

`sendmail` *Configuration File* A

A Sample `sendmail` *Configuration File*

Code Example 3-1 shows the default `main.cf` file. A description of the syntax and semantics used in this file is included in “sendmail Configuration File Syntax” on page 77 .

Code Example 3-1 The Default `main.cf` File *(1 of 11)*

```
##############################################################
#
#          Sendmail configuration file for "MAIN MACHINES"
#
#          You should install this file as /etc/sendmail.cf
#          if your machine is the main (or only) mail-relaying
#          machine in your domain. Then edit the file to
#          customize it for your network configuration.
#
#          @(#)main.mc 1.17 90/01/04 SMI

###        local info

# delete the following if you have no sendmailvars table
Lmmaildomain
# my official hostname
# You have two choices here. If you want the gateway machine to identify
# itself as the DOMAIN, use this line:
Dj$m
```

Code Example 3-1 The Default `main.cf` File *(2 of 11)*

```
# If you want the gateway machine to appear to be INSIDE the domain, use:
#Dj$w.$m
#if you are using sendmail.mx (or have a fully-qualified hostname), use:
#Dj$w

# major relay mailer - typical choice is "ddn" if you are on the
# Defense Data Network (e.g. Arpanet or Milnet)
DMsmartuucp

# major relay host: use the $M mailer to send mail to other domains
DR ddn-gateway
CR ddn-gateway

# If you want to pre-load the "mailhosts" then use a line like
# FS /usr/lib/mailhosts
# and then change all the occurrences of $%y to be $=S instead.
# Otherwise, the default is to use the hosts.byname map if NIS
# is running (or else the /etc/hosts file if no NIS).

# valid top-level domains (default passes ALL unknown domains up)
CT arpa com edu gov mil net org
CT us de fr jp kr nz il uk no au fi nl se ca ch my dk ar

# options that you probably want on a mailhost:

# checkpoint the queue after this many recipients
OC10

# refuse to send tiny messages to more than these recipients
Ob10

######################################################
#
#          General configuration information

# local domain names
#
# These can now be determined from the domainname system call.
# The first component of the NIS domain name is stripped off unless
# it begins with a dot or a plus sign.
# If your NIS domain is not inside the domain name you would like to have
# appear in your mail headers, add a "Dm" line to define your domain name.
```

Code Example 3-1 The Default `main.cf` File *(3 of 11)*

```
# The Dm value is what is used in outgoing mail. The Cm values are
# accepted in incoming mail. By default Cm is set from Dm, but you might
# want to have more than one Cm line to recognize more than one domain
# name on incoming mail during a transition.
# Example:
# DmCS.Podunk.EDU
# Cm cs cs.Podunk.EDU
#
# known hosts in this domain are obtained from gethostbyname() call

# Version number of configuration file
#ident       "@(#)version.m4       1.17       90/07/14 SMI"       /* SunOS 4.1       */
#
#          Copyright Notice
#
#Notice of copyright on this source code product does not indicate
#publication
#
#       (c) 1986,1987,1988,1989 Sun Microsystems, Inc
#          All rights reserved.


DVSMI-SVR4


###        Standard macros

# name used for error messages
DnMailer-Daemon
# special user
CDMailer-Daemon root daemon uucp
# UNIX header format
DlFrom $g  $d
# delimiter (operator) characters
Do.:%@!^=/[]
# format of a total name
Dq$g$?x ($x)$.
# SMTP login message
De$j Sendmail $v/$V ready at $b

###        Options
```

Code Example 3-1 The Default `main.cf` File *(4 of 11)*

```
# Remote mode - send through server if mailbox directory is mounted
OR
# location of alias file
OA/etc/mail/aliases
# default delivery mode (deliver in background)
Odbackground
# rebuild the alias file automagically
OD
# temporary file mode -- 0600 for secure mail, 0644 for permissive
OF0600
# default GID
Og1
# location of help file
OH/etc/mail/sendmail.hf
# log level
OL9
# default messages to old style
Oo
# Cc my postmaster on error replies I generate
OPPostmaster
# queue directory
OQ/var/spool/mqueue
# read timeout for SMTP protocols
Or15m
# status file -- none
OS/etc/mail/sendmail.st
# queue up everything before starting transmission, for safety
Os
# return queued mail after this long
OT3d
# default UID
Ou1

###        Message precedences
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100

###        Trusted users
T root daemon uucp

###        Format of headers
```

Code Example 3-1 The Default `main.cf` File *(5 of 11)*

```
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.by $j ($v/$V)
          id $i; $b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
H?x?Full-Name: $x
HSubject:
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
HErrors-To:

############################
###   Rewriting Rules  ###
############################



# Sender Field Pre-rewriting
S1
# None needed.

# Recipient Field Pre-rewriting
S2
# None needed.

# Name Canonicalization


# Internal format of names within the rewriting rules is:
#         anything<@host.domain.domain...>anything
# We try to get every kind of name into this format, except for local
# names, which have no host part. The reason for the "<>" stuff is
# that the relevant host name could be on the front of the name (for
# source routing), or on the back (normal form). We enclose the one that
# we want to route on in the <>'s to make it easy to find.
#
S3

# handle "from:<>" special case
R$*<>$*                                 $@@             turn into magic token
```

Code Example 3-1 The Default `main.cf` File *(6 of 11)*

```
# basic textual canonicalization
R<$*<@$+>>                              $@$1<@$2>
R$*<$+>$*                               $2                  basic RFC822 parsing

# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@$+,$+:$+                              @$1:$2:$3           change all "," to ":"
R@$+:$+                                 $@$>6<@$1>:$2       src route canonical
R$+:$*;@$+                              $@$1:$2;@$3         list syntax
R$+@$+                                  $:$1<@$2>           focus on domain
R$+<$+@$+>                              $1$2<@$3>           move gaze right
R$+<@$+>                                $@$>6$1<@$2>        already canonical

# convert old-style names to domain-based names
# All old-style names parse from left to right, without precedence.
R$-!$+                                  $@$>6$2<@$1.uucp>   uucphost!user
R$-.$+!$+                               $@$>6$3<@$1.$2>     host.domain!user
R$+%$+                                  $@$>3$1@$2          user%host

# Final Output Post-rewriting
S4
R$+<@$+.uucp>                           $2!$1               u@h.uucp => h!u
R$+                                     $: $>9 $1           Clean up addr
R$*<$+>$*                               $1$2$3              defocus

# Clean up an name for passing to a mailer
# (but leave it focused)
S9
R$=w!@                                  $@$w!$n
R@                                      $@$n                handle <> error addr
R$*<$*LOCAL>$*                          $1<$2$m>$3          change local info
R<@$+>$*:$+:$+                          <@$1>$2,$3:$4       <route-addr> canonical

########################
#   Rewriting rules

# special local conversions
S6
R$*<@$*$=m>$*               $1<@$2LOCAL>$4                  convert local domain

# Local and Program Mailer specification
```

Code Example 3-1 The Default `main.cf` File *(7 of 11)*

```
Mlocal,P=/usr/lib/mail.local, F=flsSDFMmnP, S=10, R=20, A=mail.local -d $u
Mprog, P=/bin/sh, F=lsDFMeuP, S=10, R=20, A=sh -c $u

S10
# None needed.

S20
# None needed.

#ident       "@(#)etherm.m4       1.15       93/04/05 SMI"       /* SunOS 4.1 */
#
#          Copyright Notice
#
#Notice of copyright on this source code product does not indicate
#publication
#
#      (c) 1986,1987,1988,1989 Sun Microsystems, Inc
#          All rights reserved.

###############################################################
#####
#####              Ethernet Mailer specification
#####
#####       Messages processed by this configuration are assumed to remain
#####       in the same domain. This really has nothing particular to do
#####               with Ethernet - the name is historical.

Mether,P=[TCP], F=msDFMuCX, S=11, R=21, A=TCP $h
S11
R$*<@$+>$*               $@$1<@$2>$3                    already ok
R$=D                     $@$1<@$w>                      tack on my hostname
R$+                      $@$1<@$k>                      tack on my mbox hostname

S21
R$*<@$+>$*               $@$1<@$2>$3                    already ok
R$+                      $@$1<@$k>                      tack on my mbox hostname


#############################################################
#   General code to convert back to old style UUCP names
S5
R$+<@LOCAL>                 $@ $w!$1                    name@LOCAL => sun!name
```

Code Example 3-1 The Default `main.cf` File *(8 of 11)*

```
R$+<@$-.LOCAL>              $@ $2!$1                    u@h.LOCAL => h!u
R$+<@$+.uucp>               $@ $2!$1                    u@h.uucp => h!u
R$+<@$*>                    $@ $2!$1                    u@h => h!u
# Route-addrs do not work here. Punt til uucp-mail comes up with something.
R<@$+>$*                    $@ @$1$2                    just defocus and punt
R$*<$*>$*                   $@ $1$2$3                   Defocus strange stuff

#          UUCP Mailer specification

Muucp,  P=/usr/bin/uux, F=msDFMhuU, S=13, R=23,
        A=uux - -r -a$f $h!rmail ($u)

# Convert uucp sender (From) field
S13
R$+                         $:$>5$1                     convert to old style
R$=w!$+                     $2                          strip local name
R$+                         $:$w!$1                     stick on real host name

# Convert uucp recipient (To, Cc) fields
S23
R$+                         $:$>5$1                     convert to old style

#ident       "@(#)ddnm.m4       1.8       93/06/30 SMI"       /* SunOS 4.1       */
#
#          Copyright Notice
#
#Notice of copyright on this source code product does not indicate
#publication
#
#      (c) 1986,1987,1988,1989 Sun Microsystems, Inc
#          All rights reserved.


###############################################################
#
#                 DDN Mailer specification
#
#          Send mail on the Defense Data Network
#                       (such as Arpanet or Milnet)

Mddn,P=[TCP], F=msDFMuCX, S=22, R=22, A=TCP $h, E=\r\n
```

Code Example 3-1 The Default `main.cf` File *(9 of 11)*

```
# map containing the inverse of mail.aliases
# Note that there is a special case mail.byaddr will cause reverse
# lookups in both Nis+ and NIS.
# If you want to use ONLY Nis+ for alias inversion comment out the next line
# and uncomment the line after that
DZmail.byaddr
#DZREVERSE.mail_aliases.org_dir

S22
R$*<@LOCAL>$*              $:$1
R$-<@$->                   $:$>3${Z$1@$2$}             invert aliases
R$*<@$+.$*>$*              $@$1<@$2.$3>$4              already ok
R$+<@$+>$*                 $@$1<@$2.$m>$3              tack on our domain
R$+                        $@$1<@$w.$m>                tack on our full name

# "Smart" UUCP mailer: Uses UUCP transport but domain-style naming
Msmartuucp, P=/usr/bin/uux, F=CmsDFMhuU, S=22, R=22,
          A=uux - -r $h!rmail ($u)


###################################################################
#
#                RULESET ZERO
#
#          This is the ruleset that determines which mailer a name goes to.

# Ruleset 30 just calls rulesets 3 then 0.
S30
R$*                      $: $>3 $1                   First canonicalize
R$*                      $@ $>0 $1                   Then rerun ruleset 0

S0
# On entry, the address has been canonicalized and focused by ruleset 3.
# Handle special cases.....
R@                       $#local $:$n                handle <> form

# resolve the local hostname to "LOCAL".
R$*<$*$=w.LOCAL>$*       $1<$2LOCAL>$4               thishost.LOCAL
R$*<$*$=w.uucp>$*       $1<$2LOCAL>$4                thishost.uucp
R$*<$*$=w>$*              $1<$2LOCAL>$4              thishost

# Mail addressed explicitly to the domain gateway (us)
```

Code Example 3-1 The Default `main.cf` File *(10 of 11)*

```
R$*<@LOCAL>             $@$>30$1                    strip our name, retry
R<@LOCAL>:$+             $@$>30$1                   retry after route strip

# For numeric spec, you can't pass spec on to receiver, since old rcvr's
# are not smart enough to know that [x.y.z.a] is their own name.
R<@[$+]>:$*              $:$>9 <@[$1]>:$2           Clean it up, then...
R<@[$+]>:$*              $#ether $@[$1] $:$2        numeric internet spec
R<@[$+]>,$*              $#ether $@[$1] $:$2        numeric internet spec
R$*<@[$+]>              $#ether $@[$2] $:$1         numeric internet spec

# deliver to known ethernet hosts explicitly specified in our domain
R$*<@$%y.LOCAL>$*       $#ether $@$2 $:$1<@$2>$3    user@host.sun.com
# deliver to hosts in our domain that have a MX record
R$*<@$%x.LOCAL>$*       $#ether $@$2 $:$1<@$2>$3    user@host.sun.com

# etherhost.uucp is treated as etherhost.$m for now.
# This allows them to be addressed from uucp as foo!sun!etherhost!user.
R$*<@$%y.uucp>$*       $#ether $@$2 $:$1<@$2>$3     user@etherhost.uucp

# Explicitly specified names in our domain -- that we've never heard of
R$*<@$*.LOCAL>$*      $#error $:Never heard of host $2 in domain $m

# Clean up addresses for external use -- kills LOCAL, route-addr ,=>:
R$*                  $:$>9 $1                       Then continue...

# resolve UUCP-style names
R<@$-.uucp>:$+        $#uucp  $@$1 $:$2             @host.uucp:...
R$+<@$-.uucp>         $#uucp $@$2 $:$1              user@host.uucp

# Pass other valid names up the ladder to our forwarder
#R$*<@$*.$=T>$*       $#$M    $@$R $:$1<@$2.$3>$4   user@domain.known

# Replace following with above to only forward "known" top-level domains
R$*<@$*.$+>$*         $#$M   $@$R $:$1<@$2.$3>$4    user@any.domain

# if you are on the DDN, then comment-out both of the lines above
# and use the following instead:
#R$*<@$*.$+>$*        $#ddn $@ $2.$3 $:$1<@$2.$3>$4  user@any.domain

# All addresses in the rules ABOVE are absolute (fully qualified domains).
# Addresses BELOW can be partially qualified.
```

Code Example 3-1 The Default `main.cf` File *(11 of 11)*

```
# deliver to known ethernet hosts
R$*<@$%y>$*           $#ether $@$2 $:$1<@$2>$3          user@etherhost
# deliver to known ethernet hosts that has MX record
R$*<@$%x>$*           $#ether $@$2 $:$1<@$2>$3          user@etherhost

# other non-local names have nowhere to go; return them to sender.
R$*<@$+.$->$*         $#error $:Unknown domain $3
R$*<@$+>$*            $#error $:Never heard of $2 in domain $m
R$*@$*                $#error $:I don't understand $1@$2

# Local names with % are really not local!
R$+%$+                $@$>30$1@$2                       turn % => @, retry

# everything else is a local name
R$+                   $#local $:$1                      local names
```

A

`sendmail` *Options*

`sendmail` *Command-Line Arguments*

Use command-line arguments on the `/usr/lib/sendmail` command line. These arguments are described in Table B-1.

Table B-1 `sendmail` Command-Line Arguments

Argument	Description
`-B`*type*	Select the body type (7BIT or 8BITMIME).
`-b`*x*	Set operation mode to *x*; operation modes are:
	`a` Run in ARPANET mode.
	`d` Run as a daemon.
	`i` Initialize the alias database.
	`m` Deliver mail (default).
	`p` Print the mail queue.
	`s` Use SMTP on input side.
	`t` Run in test mode.
	`v` Just verify recipients.
`-C`*file*	Use a different configuration file.
`-d`*level*	Set debugging level.

Table B-1 `sendmail` Command-Line Arguments *(Continued)*

Argument	Description
`-F`*name*	Set the full name of this user to *name*.
`-f`*name*	An obsolete form of `-r`.
`-h`*cnt*	Set the "hop count" to *cnt*. It sets the number of times this message has been processed by `sendmail` (to the extent that it is supported by the underlying networks). *cnt* is incremented during processing, and if it reaches the value of configuration option `h`, `sendmail` returns the message with an error.
`-M` *id*	Attempt to deliver the queued with the message-id *id*.
`-n`	Do not do aliasing or forwarding.
`-o`*xvalue*	Set configuration option *x* to the specified *value*.
`-p`*protocol*	Set the sending protocol; the *protocol* field can be in entered as *protocol*:*host* to set both the protocol and the sending host.
`-q`*time*	Try to process the queued mail. If the time is given, `sendmail` repeatedly runs through the queue at the specified interval to deliver queued mail; otherwise, it runs only once.
`-q`*Xstring*	Run the queue once, limiting the jobs to those matching *Xstring*; the letter *X* can be: `I` Limit based on queue indentifier (see -M). `R` Limit based on the recipient (see -R). `S` Limit based on the sender.
`-R`*string*	Attempt to deliver any message with a recipient containing *string*.
`-t`	Read the header for `To:`, `Cc:`, and `Bcc:` lines, and send to everyone listed in those lists. The `Bcc:` line is deleted before sending. Any names in the argument vector are deleted from the send list.
`-v`	Use verbose mode.
`-X`*logfile*	Log all traffic in and out of `sendmail` in the indicated *logfile*.

These options are described in the next section, "sendmail Configuration Options."

You can specify several configuration options as primitive flags. These are the `c`, `e`, `i`, `m`, `T`, and `v` arguments. Also, you can specify the `f` configuration option as the `-s` argument.

`sendmail` *Configuration Options*

You can set the options shown in Table B-2 using the `-o` flag on the command line or the `O` line in the configuration file.

Table B-2 `sendmail` Configuration Options *(1 of 5)*

Option	Description
A*file*	Use the named *file* as the alias file instead of `/etc/mail/aliases`. If no file is specified, use aliases in the current directory.
a*time*	Wait a set amount of time (in minutes) for an `@:@` entry to exist in the alias database before starting up. If it does not appear after that time, rebuild the database.
B*value*	Blank substitute. Default is the dot (`.`) character.
b*n*	Disallow empty messages to more than *n* recipients.
C*n*	Check after *n* recipients.
c	If an outgoing mailer is marked as being expensive, do not connect immediately. A queue process must be run to actually send the mail.
D	If set, rebuild the alias database if necessary and possible. If this option is not set, `sendmail` never rebuilds the alias database unless explicitly requested with `-bi`.
d*x*	Deliver in mode *x*. Legal modes are: `i` Deliver interactively (synchronously). `b` Deliver in background (asynchronously). `q` Queue the message (deliver during queue run).
E*string*	Append error messages with *string*; if *string* starts with a slash, it is assumed to be the pathname of a file containing a message.
e*x*	Dispose of errors using mode *x*. The values for *x* are: `p` Print error messages (default).

Table B-2 `sendmail` Configuration Options *(2 of 5)*

Option	Description
	`q` No messages, just give exit status.
	`m` Mail back errors to sender.
	`w` Write back errors (mail if user not logged in).
	`e` Mail back errors and always give zero exit status.
`F`*n*	The temporary queue file mode, in octal. Values of `644` and `600` are good choices for *n*.
`f`	Save UNIX-style `From` lines at the front of headers; normally they are assumed to be redundant and discarded.
`g`*n*	Set the default group ID for mailers to run in to *n*.
`H`*file*	Specify the help file for SMTP [Postel 82].
`h`*n*	Set maximum hop count to *n*.
`I`	Insist that the name server be running to resolve host requests.
`i`	Ignore dots in incoming messages.
`J`*path*	Set the path for searching for users .forward files.
`j`	Send error messages in MIME format.
`K`*timeout*	Set the maximum amount of time a cached connection will be permitted to be idle.
`k`*n*	Select the maximum number of open connections that will be cached at a time. The default is 1.
`L`*n*	Set the default log level to *n*.
`l`	If there is an Errors-To: header, send the error messages to the addresses listed there.
`M`*xvalue*	Set the macro *x* to *value*; this is intended only for use from the command line.
`m`	Send to the sender also, even if the sender is in an alias expansion.
`n`	Validate the RHS of aliases when rebuilding the aliases database.
`O`*options*	Set server SMTP options. The options are *key=value* pairs. The *key* can be:

Table B-2 `sendmail` Configuration Options *(3 of 5)*

Option	Description
`Addr`	Address mask (the default value is INADDR_ANY).
`Family`	Address family (the default value is INET).
`Listen`	Size of listen queue (the default value is 10).
`Port`	Name/number of the listening port (the default value is smtp).
o	Assume that the headers may be in old format; that is, spaces delimit names. This flag actually turns on an adaptive algorithm: If any recipient name contains a comma, parenthesis, or angle bracket, it is assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.
P*name*	Send the header from error messages from the `MAILER-DAEMON` to this name (which is the local postmaster).
p*opt*`,` *opt*	Set privacy options. The value for *opt* can be:
`authwarnings`	Put X-Authentication-Warning: headers in messages.
`goaway`	Disallow SMTP status queries.
`needexpnhelo`	Insist on HELO or EHLO command before EXPN.
`needmailhelo`	Insist on HELO or EHLO command before MAIL.
`needvrfyhelo`	Insist on HELO or EHLO command before VRFY.
`noexpn`	Disallow EXPN.
`novrfy`	Disallow VRFY.
`public`	Allow open access.
`restrictmailq`	Restrict `mailq` command.
`restrictqrun`	Restrict -q command line flag to root and the owner of the queue.
Q*dir*	Use the named *dir* as the queue directory.
q*factor*	Use factor as the multipler in the map function to decide when to just queue up jobs rather than run then; defaults to 600000.

Table B-2 `sendmail` Configuration Options *(4 of 5)*

Option	Description
r*timeouts*	Timeout reads after an interval.The timeouts argument is a list of *keyword=value* pairs.The recognized timeouts, their default and minimum values are:
`command`	Command read [1h, 3m].
`datablock`	Data block read [1h, 3m].
`datafinal`	Reply to a final "." in data [1h, 10m].
`datainit`	Reply to DATA command [5m, 2m].
`helo`	Reply to HELO or EHLO commands [5m, none].
`ident`	IDENT protocol timeout [30s, none].
`initial`	.Wait for initial greeting message [5m, 5m].
`mail`	Reply to MAIL command [10m, 5m].
`misc`	Reply to NOOP or VERB commands [2m, none].
`quit`	Reply to QUIT command [2m, none].
`rcpt`	Reply to RCPT command [1h, 5m].
`rset`	Reply to RSET command [5m, none].
S*file*	Save statistics in the named *file*.
s	Always initiate the queue file, even if you are going to try immediate delivery. `sendmail` always initiates the queue file before returning control to the client under any circumstance.
T*rtime/wtime*	Set the queue timeout to *rtim*e. After this interval, messages that have not been successfully sent are returned to the sender. The *wtime* variable is optional and selects the time after which a warning message is sent.
t*tzinfo*	Set the time zone.
u*n*	Set the default user ID for mailers to *n*. Mailers without the `S` flag in the mailer definition are run as this user.
V*fallbackhost*	Select *fallbackhost* to act like a low priority MX on every host.
v	Run in verbose mode.

Table B-2 `sendmail` Configuration Options *(5 of 5)*

Option	Description
`w`	Connect directly to a host as though it has no MX records at all. This option is not recommended.
`X`*n*	Set the load average value, so that the `sendmail` daemon refuses incoming SMTP connections when the system is overloaded to reduce system load. The default is 12, 0 disables this feature.
`x`*n*	Set the load average value so that `sendmail` simply queues mail (regardless of the `d`*x* option) to reduce system load. Default is 8, 0 disables this feature.
`Y`	Deliver each job that is run from the queue in a separate process.
`y`*factor*	The *factor* is added to the priority for each recipient (this lowering the priority of the job for messages with many recipients). The default is 30000.
`Z`*factor*	The *factor* is added to the priority for each recipient (this lowering the priority of the job for messages with many recipients). The default is 90000.
`z`*factor*	The *factor* is added to the priority every time a job is processed. The default is 1800.
`7`	Strip input to seven bits for compatibility with old systems.

Mailer Flags

The flags you can set in the mailer description are described in Table B-3.

Table B-3 `sendmail` Flags Set in the Mailer Description

Flag	Description
`a`	Run Extended SMTP protocol
`b`	Force a blank line at the end of a message.
`C`	Append the `@domain` clause from the sender to any names in the header that do not have an at sign (`@`) after being rewritten by rule set 3. This option is not recommended. This flag allows mail with headers with this form: `From:` *user*`1@`*local* `To:` *user*`2,` *user*`3@`*remote* to be automatically rewritten as:

Table B-3 `sendmail` Flags Set in the Mailer Description *(Continued)*

Flag	Description
	`From:` *user*`1@`*local* `To:` *user*`2@`*local*`,` *user*`3@`*remote*
`c`	D not include comments in addresses.
`D`	Look for a `Date:` header line.
`E`	Escape `From` lines to be `>From` (usually specified with `U`).
`e`	Avoid connecting to this mailer, which is expensive, normally; any necessary connection occurs during a queue run.
`F`	Look for a `From:` header line.
`f`	Look for an `-f` *from* flag, but only if this is a network forward operation (that is, the mailer gives an error if the executing user does not have special permissions).
`g`	Send error messages from the MAILER-DAEMON instead of using the null return address.
`h`	Preserve uppercase in host names for this mailer.
`I`	Select SMTP when contacting another `sendmail`.
`L`	Limit the line lengths as specified in RFC 821.
`l`	Perform final delivery because this mailer is local.
`M`	Look for a `Message-Id:` header line.
`m`	Enable the mailer to send a message to multiple users on the same host in one transaction. When a `$u` macro occurs in the `argv` part of the mailer definition, that field is repeated as necessary for all qualifying users. The `L=` field of the mailer description can be used to limit the total length of the `$u` expansion.
`n`	Do not insert a UNIX-style `From` line on the front of the message.
`P`	Look for a `Return-Path:` line.
`p`	Always add local host name to the `MAIL From:` line of SMTP, even if there already is one.
`r`	Send a `-r` flag. Performs the same function as `-f`.
`S`	Do not reset the user ID before calling the mailer. This flag would be used in a secure environment where `sendmail` ran as root. This flag could be used to avoid forged names.

Table B-3 `sendmail` Flags Set in the Mailer Description *(Continued)*

Flag	Description
`s`	Strip quote characters off the name before calling the mailer.
`U`	Look for UNIX-style `From` lines with the UUCP-style `remote from` <*host*> on the end.
`u`	Preserve uppercase in user names for this mailer.
`X`	Use the hidden dot algorithm as specified in RFC 821; basically, any line beginning with a dot will have an extra dot appended (to be stripped at the other end). This flag ensures that lines in the message containing a dot do not terminate the message prematurely.
`x`	Look for a `Full-Name:` header line.
`7`	Strip output to seven bits.

B

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

N

O

P

Q

R

S

T

U

V

W

X

Y

Z