SCO

SCO OpenServer™ Mail and Messaging Guide

SCO OpenServer™

SCO OpenServer™
Mail and Messaging Guide

Document Version: 5.0.5
1 August 1998

About this book 1

Chapter 1
Using and administering electronic mail 7

Chapter 2
Using e-mail 13

About this book

This book provides instructions for using the electronic mail programs **mail**(C) and **SCO Shell Mail** (for more information on SCO Shell, see the *Operating System User's Guide*). It includes instructions for mail system configuration and administration using either **sendmail** or MMDF.

Mail administrators should be familiar with basic SCO® system operation. Documentation for the Desktop mail program, **scomail**, is provided online in *Using Mail.*

You will find the information you need more quickly if you are familiar with:

- "How this book is organized" (this page)
- "Related documentation" (page 2)
- "Typographical conventions" (page 5)

Although we try to present information in the most useful way, you are the ultimate judge of how well we succeed. Please let us know how we can improve this book (page 6).

How this book is organized

After introducing SCO mail systems, this book describes how to:

- use two Mail User Agents for sending and receiving electronic mail:
  - **mail** (page 13)
  - **SCO Shell Mail** (page 39)
- administer a mail system, including:
  - using the **MMDF Configuration Manager** (page 68) to configure a system that uses MMDF for mail transfer

- using the **MMDF administration managers** (page 68) to administer a system that uses MMDF
- administering a local (page 221) MMDF mail system on a machine that is not connected to a network
- administering a system (page 149) that uses **sendmail**, including standard and advanced administrative tasks
- tuning a system (page 153) that uses **sendmail**

Related documentation

SCO OpenServer™ systems include comprehensive documentation. Depending on which SCO OpenServer system you have, the following books are available in online and/or printed form. Access online books by double-clicking on the Desktop **Help** icon. Additional printed versions of the books are also available. The Desktop and most SCO OpenServer programs and utilities are linked to extensive context-sensitive help, which in turn is linked to relevant sections in the online versions of the following books. See "Getting help" in the *SCO OpenServer Handbook*.

NOTE When you upgrade or supplement your SCO OpenServer software, you might also install online documentation that is more current than the printed books that came with the original system. For the most up-to-date information, check the online documentation.

Release Notes
: contain important late-breaking information about installation, hardware requirements, and known limitations. The *Release Notes* also highlight the new features added for this release.

SCO OpenServer Handbook
: provides the information needed to get your SCO OpenServer system up and running, including installation and configuration instructions, and introductions to the Desktop, online documentation, system administration, and troubleshooting.

Graphical Environment Guide
: describes how to customize and administer the Graphical Environment, including the X Window System™ server, the SCO® Panner™ window manager, the Desktop, and other X clients.

Graphical Environment help
: provides online context-sensitive help for Calendar, Edit, the Desktop, Help, Mail, Paint, the SCO Panner window manager, and the UNIX® command-line window.

Graphical Environment Reference
contains the manual pages for the X server (section X), the Desktop, and X clients from SCO and MIT (section XC).

Guide to Gateways for LAN Servers
describes how to set up SCO® Gateway for NetWare® and LAN Manager Client software on an SCO OpenServer system to access printers, file-systems, and other services provided by servers running Novell® NetWare® and by servers running LAN Manager over DOS, OS/2®, or UNIX systems. This book contains the manual pages for LAN Manager Client commands (section LMC).

Networking Guide
provides information on configuring and administering TCP/IP, NFS®, and IPX/SPX™ software to provide networked and distributed functionality, including system and network management, applications support, and file, name, and time services.

Networking Reference
contains the command, file, protocol, and utility manual pages for the IPX/SPX (section PADM), NFS (sections NADM, NC, and NF), and TCP/IP (sections ADMN, ADMP, SFF, and TC) networking software.

Operating System Administrator's Reference
contains the manual pages for system administration commands and utilities (section ADM), system file formats (section F), hardware-specific information (section HW), miscellaneous commands (section M), and SCO Visual Tcl™ commands (section TCL).

Operating System Tutorial
provides a basic introduction to the SCO OpenServer operating system. This book can also be used as a refresher course or a quick-reference guide. Each chapter is a self-contained lesson designed to give hands-on experience using the SCO OpenServer operating system.

Operating System User's Guide
provides an introduction to SCO OpenServer command-line utilities, the SCO Shell utilities, working with files and directories, editing files with the **vi** editor, transferring files to disks and tape, using DOS disks and files in the SCO OpenServer environment, managing processes, shell programming, regular expressions, **awk**, and **sed**.

Operating System User's Reference
contains the manual pages for user-accessible operating system commands and utilities (section C).

PC-Interface Guide
: describes how to set up PC-Interface™ software on an SCO OpenServer system to provide print, file, and terminal emulation services to computers running PC-Interface client software under DOS or Microsoft® Windows™.

Performance Guide
: describes performance tuning for uniprocessor, multiprocessor, and networked systems, including those with TCP/IP, NFS, and X clients. This book discusses how the various subsystems function, possible performance constraints due to hardware limitations, and optimizing system configuration for various uses. Concepts and strategies are illustrated with case studies.

SCO Merge User's Guide
: describes how to use and configure an SCO® Merge™ system. Topics include installing Windows, installing DOS and Windows applications, using DOS with the SCO OpenServer operating system, configuring hardware and software resources, and using SCO Merge in an international environment.

SCO Wabi User's Guide
: describes how to use SCO® Wabi™ software to run Windows 3.1 applications on the SCO OpenServer operating system. Topics include installing the SCO Wabi software, setting up drives, configuring ports, managing printing operations, and installing and running applications.

System Administration Guide
: describes configuration and maintenance of the base operating system, including account, filesystem, printer, backup, security, UUCP, and virtual disk management.

The SCO OpenServer Development System includes extensive documentation of application development issues and tools.

Many other useful publications about SCO systems by independent authors are available from technical bookstores. For mail, these include:

general
: *!%@:: A Directory of Electronic Mail Addressing and Networks.* Donnalyn Frey & Rick Adams, O'Reilly & Associates, Inc., 1993. ISBN: 1-56592-031-7.

  Delivering Electronic Mail. Phillip Robinson, M&T Books, 1992. ISBN: 1-55851-170-9.

sendmail
: *sendmail.* Costales, Bryan, with Eric Allman & Neil Rickert, O'Reilly & Associates, Inc., 1993. ISBN: 1-56592-056-2.

Typographical conventions

This publication presents commands, filenames, keystrokes, and other special elements as shown here:

Example:	Used for:
lp or **lp**(C)	commands, device drivers, programs, and utilities (names, icons, or windows); the letter in parentheses indicates the reference manual section in which the command, driver, program, or utility is documented
/new/client.list	files, directories, and desktops (names, icons, or windows)
root	system, network, or user names
filename	placeholders (replace with appropriate name or value)
⟨Esc⟩	keyboard keys
`Exit program?`	system output (prompts, messages)
yes or **yes**	user input
"Description"	field names or column headings (on screen or in database)
Cancel	button names
Edit	menu names
Copy	menu items
File ⇨ Find ⇨ Text	sequences of menus and menu items
open or **open**(S)	library routines, system calls, kernel functions, C keywords; the letter in parentheses indicates the reference manual section in which the file is documented
$HOME	environment or shell variables
SIGHUP	named constants or signals
"adm3a"	data values
employees	database names
orders	database tables
`buf`	C program structures
`b_b.errno`	structure members

How can we improve this book?

What did you find particularly helpful in this book? Are there mistakes in this book? Could it be organized more usefully? Did we leave out information you need or include unnecessary material? If so, please tell us.

To help us implement your suggestions, include relevant details, such as book title, section name, page number, and system component. We would appreciate information on how to contact you in case we need additional explanation.

To contact us, use the card at the back of the *SCO OpenServer Handbook* or write to us at:

Technical Publications
Attn: CFT
The Santa Cruz Operation, Inc.
PO Box 1900
Santa Cruz, California 95061-9969
USA

or e-mail us at:

techpubs@sco.com or ... *uunet!sco!techpubs*

Thank you.

Chapter 1

Using and administering electronic mail

Your SCO OpenServer™ system provides a variety of electronic mail options, including:

- a choice of Mail User Agents (this page) including the addition of the **scosh**(C) mail reader to the SCO OpenServer Desktop System
- a choice of Mail Transfer Agents (page 8)
- interoperability with two DOS messaging systems (page 10) as well as other UNIX systems
- support for mail exchange using the POP and IMAP servers (page 11)
- graphical utilities for configuring and administering MMDF (page 11)
- support for graphical, audio, video, and multimedia messages (page 12)

Mail User Agents

The SCO system MAIL package includes a variety of Mail User Agents (MUAs):

mail(C)	the operating system's command-line based message processor
scomail(XC)	the graphical Desktop message processor
SCO Shell Mail	the character-based mailer designed for the SCO menu-based shell, **scosh**(C)

Individual users on a system can choose their own MUA.

Mail Transfer Agents

A Mail Transfer Agent (MTA), or mail router, accepts messages generated by a Mail User Agent, determines a route for delivery, edits the message header as required by the destination and delivery program, and calls the appropriate delivery program to deliver the mail.

You can choose between two Mail Transfer Agents, provided with SCO systems, **mmdf**(ADM) and **sendmail**(ADMN). The default MTA is MMDF, but you can choose to switch to **sendmail** either at installation time or subsequently. See "Changing Mail Transfer Agents" (page 9).

NOTE **sendmail**(ADMN) has been updated to version 8.8.8.

Comparison of sendmail with MMDF

Both of these MTAs:

- provide a general inter-network mail routing facility with automatic routing to network gateways, message batching, queueing, and retransmission.
- support either Internet-style addressing (*user@domain*) or UUCP-style addressing (*host!user*).
- work with delivery agents such as SMTP (Simple Mail Transfer Protocol), X.400, and UUCP. X.400 support is available separately from SCO.
- provide aliasing and forwarding capability.
- support customized mailers.
- provide flexible configuration.
- support routing of messages that conform to the Multipurpose Internet Mail Extensions (MIME) standard.

MMDF supports two-stage timeout, which **sendmail** does not support. MMDF uses two-stage timeout when routing mail through machines to users. If a message cannot be forwarded to a particular machine or to a particular user on a machine, a warning is sent back to the mail message sender. This is stage one. At some future time (configurable by the administrator), the message is relayed again. If it fails, a failure message is returned to the sender, and MMDF makes no further attempts to resend the original message. This is stage two.

MMDF offers several substantial benefits over **sendmail**, including:

- Graphical utilities for configuration and administration.
- Configuration files that are easy to read and understand.
- The ability for end-users to configure their own sorting parameters.
- A larger set of supported delivery agents.

Because MMDF does not consider backwards compatibility a design goal, the address parsing is simpler but much less flexible.

It is somewhat more difficult to integrate a new delivery agent ("channel") into MMDF. In particular, MMDF must know the location and format of host tables for all channels, and each channel must speak a special protocol. This allows MMDF to do additional verification (such as verifying host names) at submission time.

MMDF strictly separates the submission and delivery phases. **sendmail** understands each of these stages, but they are integrated into one program.

sendmail is difficult to configure manually. This implementation of **sendmail**, however, includes a script that creates a basic configuration that is adequate for most sites.

See also:

- Chapter 4, "Managing mail with MMDF" (page 67)
- Chapter 5, "sendmail administration" (page 149)

Changing Mail Transfer Agents

MMDF and **sendmail** are the two MTAs supplied with SCO systems. A choice is made at installation time, but can be changed later. If your mail system is complex, changing Mail Transfer Agents may be non-trivial. For the rationale of using one over the other, refer to "Comparison of sendmail with MMDF" (page 8).

To replace MMDF with **sendmail** (or the converse) as the Mail Transfer Agent:

1. run **custom**(ADM)
2. Select the Mail Transfer Agent that is currently installed, either "SCO MMDF" or "SCO SendMail".
3. Remove the component.
4. Select the Mail Transfer Agent to install, either "SCO MMDF" or "SCO SendMail".
5. Run the installation.

If you must switch between using MMDF and **sendmail** frequently on the same system, use the following procedure:

1. Use the **Software Manager** to load the "SCO SendMail" component — see "Loading software" in the *SCO OpenServer Handbook*.
2. To switch from MMDF to **sendmail** on an SCO OpenServer Enterprise System, enter the commands:

   custom -p SCO:odtes -d SCO:MMDF
   custom -p SCO:odtes -e SCO:SendMail

   To switch from **SendMail** to MMDF on an SCO OpenServer Enterprise System, enter:

   custom -p SCO:odtes -d SCO:SendMail
   custom -p SCO:odtes -e SCO:MMDF

NOTE For either of the Mail Transfer Agent changes described above, on an SCO OpenServer Host System, substitute **SCO:unixos** for **SCO:odtes**. On an SCO OpenServer Desktop System, substitute **SCO:odtps** for **SCO:odtes**.

MMDF interoperability

This release of MMDF provides interoperability with these DOS messaging systems by way of their SMTP gateways:

- cc:Mail™
- Microsoft® Mail/Windows for Workgroups

Install either of these gateways using the documentation provided with that product. Use the **Host Administration Manager** to set up mail exchange with the gateway machine over the **smtp** channel. See "Adding or removing a mail host" (page 80).

POP3 and IMAP4 servers

This release provides Post Office Protocol (POP3) server software and Internet Message Access Protocol (IMAP4) server software. For more information on IMAP, see **imapd**(ADMN).

Using the POP server

Using the POP server enables many popular mail programs on PC and Macintosh systems (including Netscape Navigator and other POP clients) to receive mail using your system as the server. Refer to your POP client documentation for instructions.

The POP3 server runs by default when the system is installed; it is listed in *inetd.conf*(SFF) and is invoked by **inetd**(ADMN) when POP requests are received.

Removing the POP server

If you want to remove the POP server software, use **custom**(ADM). In **custom**, specify the "Mail User Agents" component, then the "SCO Post Office Protocol 3 Server" package, and select **Remove**. This removes the startup instructions from */etc/inetd.conf*.

MMDF configuration and administration managers

This release of MMDF provides several new utilities for administration and configuration of electronic mail using MMDF. Mail administrators can now choose to avoid editing mail configuration files by hand for most tasks.

- The **Mail Configuration Manager** is a graphical interface for initial mail configuration. A newly installed SCO system is configured to deliver electronic mail locally (between users on the system itself). After installation, run the **Mail Configuration Manager** if you want to exchange mail over a network or if you want to redirect mail sent to special system accounts, such as *root*, *mmdf*, or *postmaster*.
- The **Host Administration Manager** is a graphical tool for ongoing administration of mail hosts.
- The **Aliases Administration Manager** is a graphical tool for administering mail aliases.
- Additional graphical managers are available for detailed administration of MMDF tables, channels, and domains. See "How to start a Mail manager" (page 68).

MIME conformance

The Multipurpose Internet Mail Extensions (MIME) standard supports graphical, audio, video, and multimedia messages. In addition, it allows inclusion of multiple message parts in a single message, allows inclusion of binary files, and supports non-ASCII character sets and multiple-font messages.

User configurable timeout

To prevent long jobs from stopping and then running again many times on the local channel, users with suitable authorization can configure the timeout to allow such jobs to complete. To set the timeout, add the following line to the **MCHN** entry for the local channel in */usr/mmdf/mmdftailor*:

```
confstr="timeout=600"
```

Chapter 2

Using e-mail

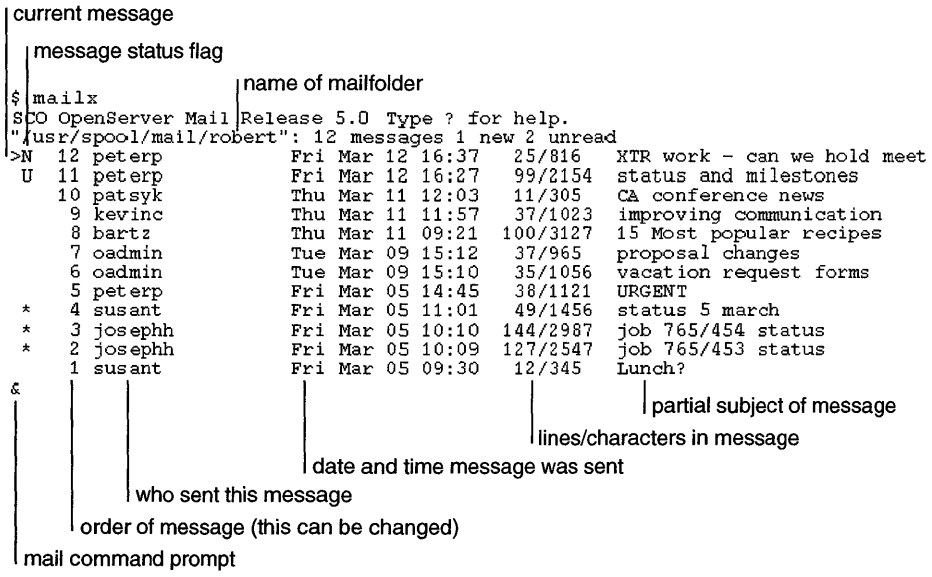
You can exchange mail with any of the users on your computer. If your computer is linked to others via a network, you can also exchange mail with users on the other machines.

This chapter describes the command-line version of the **mail**(C) program provided with SCO systems. For information on the menu-driven **SCO Shell Mail** program, see Chapter 3, "Using SCO Shell Mail" (page 39). The graphical Desktop **Mail** program is described online in *Using Mail.*

E-mail is like a postal system, and the system mailbox is like the post office. The system mailbox is located in the *usr/spool/mail* directory, and contains a file for each user on your system. Your own personal or user mailbox is the file named *mbox* in your home directory. Mail sent to you is put in your system mailbox and is automatically saved in your user mailbox after you have read it. For further information on default mail filenames and locations, see the **mail**(C) manual page. For details on controlling certain aspects of mail using environment variables, see "Setting environment variables" in the *Operating System User's Guide.*

Starting and quitting mail

To start **mail**(C), type **mail** on the UNIX command line and press ⟨Enter⟩. The opening screen shows the headers of the messages waiting in your system mailbox:



As new messages arrive, they are added to the message list. The greater-than sign (>) is a status flag that shows the current message. To change the order in which headers are displayed, see "Changing the order of the header list" (page 17).

Here is a complete list of message status flags:

Symbol	Description
>	current message
U	unread message
N	new message
M	message saved to user mailbox using **mbox** command
*	message saved to a file using **save** or **write** commands
P	message preserved (using **hold** command) in your system mailbox instead of being saved automatically to your user mailbox

If there are no messages in your system mailbox, **mail** exits immediately with a message. When your mailbox is empty, you can only run **mail** to create a new message (page 19) or to read a different mail folder (page 26).

Getting help

To get help when you are reading your mail, type **?** at the **mail** prompt. To get help while you are inputing a message, go to a new line and enter **~?**.

Quitting mail

To quit **mail** and update your mailbox, type **q** at the **mail** prompt.

If you want to leave **mail** quickly without altering either your system or user mailbox, use the **exit** (**x**) command. This leaves **mail** without changing anything.

Reading mail

To read a message, type its number and press ⟨Enter⟩. The message is then displayed. You see the message header (including "To", "From", and "Subject" lines) along with the message itself:

```
& 1
Message  1:
From: susant Fri Mar 05 09:30:12 1993
From: susant@bps.com (Susan Tzarcha)
To: perry
Subject: Lunch?
Date: Fri, 05 Mar 1993 09:29:21 (EST)
Message-ID:  <9303120929.aa0915@machina.bps.com>
Status: RO

Hi, what are you doing for lunch today? Alison and Greg have
suggested we try the new restaurant in the Plaza.

&
```

Once you have read a message, it is automatically saved to your *mbox* file (by default).

Depending on how **mail** is set up, long messages may scroll off the screen. If this happens, press ⟨Ctrl⟩**S** to temporarily stop the scrolling, and ⟨Ctrl⟩**Q** to start it again. A more practical solution is to set the number of lines the screen scrolls by when you read a message. To do this, type **set crt=20** at the **mail** prompt. The screen will then pause every 20 lines. To make this setting permanent, add the line **set crt=20** to the *.mailrc* file in your home directory.

For details on setting variables in the *.mailrc* file, see "Setting environment variables" in the *Operating System User's Guide*.

You can select the program that **mail** uses to view long messages by setting the **PAGER** variable to the pathname of a suitable command (such as */usr/bin/pg*). You must also have set **crt** to a suitable value for this to work. (A paging program allows you to quit reading a message at any stage. Depending on your choice of program, it may also allow you to search for text strings and change your position in the message.)

Viewing the header list

You can redisplay the message headers using the command **h** (short for **header**). If you want to stop reading a long message to return to the headers, press ⟨Del⟩ then enter **h** to redisplay the headers. If you have more than one screen of headers, type **h+** to move forward to the next screen, and **h-** to go back to the previous one. You can also specify the number of screens to move after both + and -.

Selecting particular messages

To display the headers of messages on a particular subject, type **h/*subject*** at the **mail** prompt, where ***subject*** is all or part of the subject line you want to find.

To show only those messages from one user, type **h *login***, where ***login*** is their login name. To display these messages, one after another, enter **p *login***. For the example introduced earlier in this chapter, the command **h peterp** displays the headers of all messages received from *peterp*:

```
N  12 peterp   Fri Mar 12 16:37   25/816   XTR work - can we hold meet
U  11 peterp   Fri Mar 12 16:27   99/2154  status and milestones
    5 peterp   Fri Mar 05 14:45   38/1121  URGENT
```

You can specify messages by their number in the list of headers. There are also three special characters available for specifying messages:

^	the first message
.	the current message
$	the last message

For example, to list all message headers from the current to the last message, type:

h .-$

A dash (-) is used to indicate a range of characters; a list of message numbers separated by spaces indicates individual messages. For example:

h ^ . $

displays the headers of the first, current, and last messages only. For the example used earlier in "Setting environment variables" in the *Operating System User's Guide*, the screen would show:

```
 N  12 peterp   Fri Mar 12 16:37   25/816   XTR work - can we hold meet
>    1 susant   Fri Mar 05 09:30   12/345   Lunch?
```

Recall that the first message became the current message when you read it previously.

You can mix a range with individual message numbers:

h ^ 10-$

displays the first, the tenth, and all subsequent messages:

```
 N  12 peterp   Fri Mar 12 16:37   25/816   XTR work - can we hold meet
 U  11 peterp   Fri Mar 12 16:27   99/2154  status and milestones
    10 patsyk   Thu Mar 11 12:03   11/305   CA conference news
>    1 susant   Fri Mar 05 09:30   12/345   Lunch?
```

You can also create a message list by specifying the type of messages in which you are interested. Type a colon followed by one of the following letters:

d	deleted messages
n	new messages
o	old messages
r	read messages
u	unread messages

For example, to see a list of headers of the messages you have deleted, type:

h :d

Use an asterisk (*) if you need to specify all non-deleted messages. For example, to completely clean out your mailbox, use the **save** command with an asterisk and a filename to save all non-deleted messages to that file:

s * mail.old

The file that the messages are saved to is termed a "mail folder". You can use mail folders (page 26) to group together messages on different subjects.

Changing the order of the header list

You can display message headers with the most recent message at either the top or the bottom of the list.

To list message headers with the most recent message at the bottom (that is, in chronological order), type **set chron** at the **mail** prompt. To list message headers with most recent numbers at the top, type **set mchron**.

To make your setting apply for all your **mail** sessions, add the line **set chron** or **set mchron** to your *.mailrc* file. For details on setting variables in the *.mailrc* file, see "Setting environment variables" in the *Operating System User's Guide.*

Replying to a message

You can reply to a message with the **r** command. If you have just read a message, entering **r** at the **mail** prompt starts a response to that message. If you have not read any messages, typing **r** begins a response to the current message (shown by the status flag " > "). You can also reply to a particular message by typing **r *number***, where ***number*** is the message number.

When you respond to a message, **mail** automatically fills in the "To" and "Subject" fields for you:

```
& r 12
To: peterp
Subject: Re: XTR work - can we hold meeting?
```

If you did not intend to reply to a message, or you change your mind, enter **~q** on a new line to abandon the creation of the response.

r replies only to the sender of the message. If you want to respond to the sender and everyone who was copied on the original mail, use **R** instead.

If you want to change any of the header fields while you are composing your reply, go to a new line and type **~h**. You can then change any of the header details. See "Editing a message's header" (page 20).

Forwarding a message to another user

To forward the message you have just read, enter **f** followed by the logins of the people who are to receive it. You can forward any message by following **f** with the message number, and the logins. For example, to forward message 12 to *susant* and *bartz*:

f 12 susant bartz

The forwarded message is indented one tab stop to show that it is a forwarded message. If you do not want the message indented, use **F** instead of **f**.

To add a comment to a forwarded message, create a new message (as described in the next section) and include the message (***number***) to be forwarded by typing **~m *number*** on a new line. For more information, see "Including a file or a message in mail" (page 20).

Creating and sending a message

To create and send a message to someone else (with login name ***login***) on your machine, type:

mail ***login***

at the shell prompt or

m ***login***

at the **mail** prompt. Depending on how **mail** is set up, you may be prompted for a subject:

```
&m susant
Subject: New status report available today?
```

When typing your message, press ⟨Enter⟩ to start new lines. Line editing functions are available when entering text, including ⟨Ctrl⟩**U** to delete a line and ⟨Bksp⟩ to back up one character. See "Editing a message" (page 20).

To view the message you are creating (including the header fields) as it will appear when you send it, type **~p** at the start of a new line.

When you are ready to send your message, press ⟨Ctrl⟩**D** or enter a single dot " . " at the start of a new line. You may then be asked who you want the message to be copied to (the "Cc" line of a message). Press ⟨Enter⟩ if you do not want the message to be copied to anyone else.

Sending mail from the command line

If you wish to send an unedited file to someone (***login***) without invoking **mail**), enter:

mail ***login*** **<** ***filename***

at the command line. When you send a file to someone by this method, you can still specify a subject and carbon copy recipients. For example, you could send a file named *note* with the subject "Important Meeting" by entering the following command:

mail -s "Important Meeting" -c "peterp josephh" susant < note

The "To" field contains "susant", and the "Cc" field contains "peterp josephh".

Sending mail in this way can be used within shell scripts that need to send messages automatically. It can also be used to handle the output of **at**, **batch**, and **cron** jobs.

Editing a message's header

To edit the header details while you are entering a message, type ~h on a new line and make any necessary changes to each field. You can also select and edit individual fields as detailed below.

To add additional names to the "To" list, type:

~t ***login1 login2 ...***

You can name as many additional logins as you like. Note that users originally on the recipient list still receive the message; you cannot remove anyone from the recipient list with ~t. To remove a login, use ~h.

You can replace or add a "Subject" field by typing:

~s ***a new subject***

This replaces any previous subject with ***a new subject*** (you do not need to surround the text with quotation marks).

To add people to the "Cc" list, type:

~c ***login1 login2 ...***

Similarly, the following sequence adds a list of people to the "Bcc" (Blind carbon copy) list:

~b ***login1 login2 ...***

The people on this list receive a copy of the message, but are not mentioned anywhere in the message you send. Use this if you do not want certain people to know who else will be receiving the message.

Editing a message

You can enter the **vi** editor by typing ~v on a new line while creating a message, or by entering the **v** command at the **mail** prompt. Use the **VISUAL** environment variable to select an editor other than **vi**. Similarly, you can enter the editor selected using the **EDITOR** environment variable by typing ~e on a new line while creating a message.

For details on how to set environment variables, see "Setting environment variables" in the *Operating System User's Guide*.

Including a file or a message in mail

You can include a file from any directory, or any message from the system mailbox or current mail folder in mail that you create. To include a message from a different folder, you must first switch to that folder before beginning to create the new message.

To read a file named ***filename*** into your current message, enter **~r *filename*** on a new line. For example, suppose that *susant* wants to read the file *12Mar93* from her *reports* directory into the message she is currently editing. She enters **~r $HOME/reports/12Mar93** on a new line. **mail** reads the file into the message; it displays the name of the file, and the number of lines and characters in the file. The file itself is not shown:

```
~r $HOME/reports/12Mar93
"/u/susant/reports/12Mar93" 72/2404
```

To include a message that was previously sent to you, type **~m *number*,** where ***number*** is the number of the message. **mail** automatically indents the inclusion by one tab stop. If you do not want the message indented, use **~M *number*** instead.

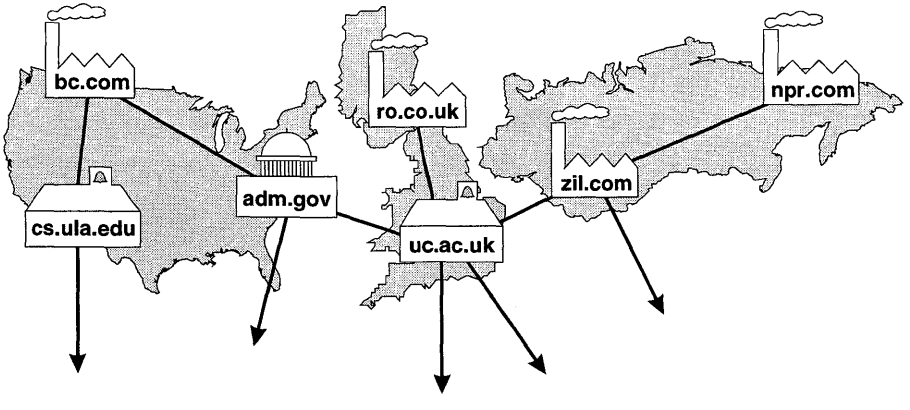
Canceling a message

To cancel a message while you are creating it, press ⟨Del⟩ twice (or **~q**). A copy of the canceled message is appended to the file *dead.letter* in your home directory. You can bring this file back into a message you are creating by typing **~d** on a blank line.

Addressing mail to users on other systems

If your SCO system is part of a network, you can send mail to users on other systems on the network. However, you need to know how to specify the name of the system to which your mail will be delivered.

You can send mail on your local network or over large area networks across an interconnected world-wide network of computer systems known as the Internet, represented here figuratively:



The Internet is divided into a series of "domains" — groups of sites identified by some collective attribute. For example, the domains in the United States include *gov* (government), *com* (commercial), *mil* (military), and *edu* (educational). Many computers in other countries are on the Internet, at sites in the United Kingdom and Russia, for example.

Each domain is divided into hierarchies of subdomains that may have an arbitrary number of levels. The lowest level subdomain is may be an individual computer name, an organization name, or the name of a division within an organization. Domains usually do not specify a machine name since this may very likely change; instead mail that arrives at an organization is redistributed locally between users on different host computers. When you receive a message from someone on a remote system, the domain of the sender appears to the right of the "@" character on the "From" line. For example:

```
From: andrea@scribe.npr.com
```

is a message from *andrea* in the subdomain *scribe* belonging to the company identified by the subdomain *npr* in the domain *com*. The whole address is known as a "fully qualified domain name".

Depending on how your system administrator has configured **mail** on your system, you only need to state the login in the "To" field. Alternatively, you may need to specify a full domain name, for example:

```
To: andrea
```

instead of

```
To: andrea@scribe.npr.com
```

You can set up aliases for such mail addresses using the **alias** command in your *.mailrc* file as described in "Making mail execute commands at startup" (page 30).

Saving a message

To save the current message including its header, to a file, enter **s** followed by the name of the "mail folder" where you want the message saved. (A mail folder is another name for a regular file containing saved mail messages. At the start and end of each message in the file is a line containing the string "^A^A^A^A"—four ASCII characters with value 1.) You can also save messages by specifying their numbers, individually, as ranges, or a combination of both. For example, **s 1-5 8 11 myfolder** saves messages 1, 2, 3, 4, 5, 8, and 11 in *myfolder*.

Each saved message is marked with an asterisk in the list of headers. When you quit from **mail**, saved messages are normally deleted from the system mailbox unless you **hold** them, or set the environment variable **keepsave**.

You can read a mail folder or your user mailbox by specifying its ***filename*** using the **-f** option:

mail -f *filename*

See "Organizing your mail into folders" (page 26).

Printing a message

To print the current message, enter **lp** at the **mail** prompt.

You can also print all the messages from a particular user, by entering **lp *login***.

Print several messages by specifying the message numbers individually, as ranges, or a mix of both.

To print your message to a printer other than the default printer for your system, first save the message (page 22) to a file, and then print the file by typing **lp -d *printer filename*** at the system prompt, or **!lp -d *printer filename*** at the **mail** prompt. This sends the file ***filename*** to the named ***printer***. For example:

!lp -d laser peters_memo

prints the saved message *peters_memo* on the printer *laser*.

Deleting a message

To delete the message you have just read, enter **d** at the **mail** prompt. If you want to delete the current message and display (print) the next on your screen, enter **dp**. You can specify several messages to be deleted by specifying their message numbers, individually, as ranges, or a mix of both. For example, **d . 3-5 12** deletes the current message, and messages 3, 4, 5, and 12.

Deleted messages are not removed from your system mailbox completely until you exit mail. They are just marked to be deleted so that the **h** command does not display their headers.

Restoring a message

You can only restore messages that you have deleted during your current **mail** session. To restore the last message you deleted, enter **u**. The restored message reappears in your message list. If you want to restore a particular message, type **u** followed by the message number(s) you want to restore. The command **u** * restores all messages that you have deleted in the current mail session.

Sending a message to a list of people using an alias

If you frequently send mail to a certain group of people (all the people in your department, for example) you can create an alias (a name given to a list of their logins), as a short way to refer to them as a group. You can use the alias when you need to address a message to them all. Treat an alias just like a user's login: put it in the "To", "Cc" or "Blind Cc" field of the message header form.

The **alias** command links a group of names with the single name given by the first argument. For example, you can create an alias called *XTRproject* by adding the following line to your *.mailrc* file:

```
alias XTRproject perry susant peterp kevinc
```

You can then use the name *XTRproject* in a destination address (**mail XTRproject**), and **mail** expands it to the four names aliased to *XTRproject*.

Aliases that you define are expanded in mail sent to others so that they can reply to each individual recipient. For example, the "To" field in a message sent to *XTRproject* reads:

```
To: perry susant peterp kevinc
```

and not:

```
To: XTRproject
```

You can find out what aliases you have set up by typing **alias** at the **mail** prompt.

You should add aliases to your *.mailrc* file. For details on setting variables in the *.mailrc* file, see "Setting environment variables" in the *Operating System User's Guide.*

Note that your system administrator may already have set up system-wide aliases for use by everyone. These aliases are not expanded when displayed in the "To" field since they are available to everyone. The file that contains the system-wide alias definitions is itself defined in the file */usr/mmdf/mmdftailor* that contains lines such as:

```
;
; Alias configuration
;
MATBL name=aliases, file="/usr/local/lib/mail/aliases.mmdf"

ALIAS table=aliases
```

These lines show that the table of aliases used by the mail system is built from the file */usr/local/lib/mail/aliases.mmdf.* (Your system may use a different pathname for the alias definitions file.) You may find that the alias definitions file is quite complicated; it can assign both aliases and logins to an alias. In this

way, it builds a hierarchy of aliases that can match the hierarchy within an organization or a project. For example, the alias definitions file might contain entries such as:

```
# teams
Ateam: alice, henry, jackie
Bteam: james, trish
BIGteam: Ateam, Bteam
```

These entries define aliases *Ateam* and *Bteam* for two groups of people. The alias *BIGteam* is composed of the members of these two aliases.

To discover who belongs to a personal or a system-wide alias, enter **A** ***alias_name*** at the **mail** prompt. **mail** expands the alias into a full list of mail addresses.

Attaching a signature to your messages

You can create a standard signature for the messages that you write. The signature serves the same purpose as the letterhead on a piece of business correspondence, and can include your name, address, business name, and other pertinent information.

If you decide to use a signature, **mail** attaches it to the end of each mail message that you send out. You do not see the signature when you create the message, but the message recipients see it.

To add a signature to your messages, add this line to your *.mailrc* file:

set sign="*your signature goes here*"

For details on setting variables in the *.mailrc* file, see "Setting environment variables" in the *Operating System User's Guide*.

An example of a signature *.mailrc* might be:

```
set sign="Brook Bond (003.5) - licensed to make tea"
```

To include your signature in a message, type **~a** at the point where you want the signature to appear.

Note that when sending mail to other sites it is considered rude to use a signature block more than four lines long; an excessively long signature costs other people money to relay it through their computers.

Adding your real name to the message header

When you read a message, the "From" field in the header information includes the sender's login (usually an abbreviation of the sender's real name).

To add your real name or other information to the "From" field, you need to change the environment variable **NAME**. For example, your login is *perry* and you want to add the information that you are a project leader. Before sending out a memorandum, you could do the following if you use the Bourne or Korn shell:

```
$ NAME="XTR project team leader"
$ export NAME
$ mail
```

In the C shell, you would enter the following:

```
$ setenv NAME "XTR project team leader"
$ mail
```

Mail received from you will now contain the line:

```
From: perry (XTR project team leader)
```

For details on how to set environment variables, see "Setting environment variables" in the *Operating System User's Guide*.

Organizing your mail into folders

Mail folders are created automatically when you save messages to a file. For example:

s 1-5 12 14 minutes

creates a mail folder called *minutes*. A mail folder is an alternate mailbox; you can do anything with it that you can do with your system mailbox.

To start **mail** from a mail folder, specify the name of the folder after **-f** option:

mail -f minutes

You can also switch folders from within **mail** with the **fold** command:

fold minutes

To see a list of your folders from within **mail**, type:

folders

If you want to keep your mail folders in a subdirectory of your home directory, you need to tell mail where to find them by adding a line **set folder=*subdirectory*** to your *.mailrc* startup file. For example, if you keep your mail folders in the subdirectory *mymail*, add the following line to *.mailrc*:

```
set folder=mymail
```

mail then looks in the directory **$HOME**/*mymail* when you use folder commands.

For details on setting variables in the *.mailrc* file, see "Setting environment variables" in the *Operating System User's Guide*.

Working with attachments and large files

SCO Shell Mail allows you to send a separate file or clipboard item along with mail messages. Unlike the message itself, this attachment can be a file with a special data format, such as a spreadsheet file, graphics file, database file, or word processing document.

Reading an attachment

If a user of **SCO Shell Mail** sends you a message, and attaches a file or clipboard item to it, the attachment is included in the message when you read it. This is because **mail**, unlike SCO Shell, does not treat an attachment as being a separate item. The attachment itself is in an encoded form that you can read by piping it through the filter **uudecode**, or by saving the message in a file and decoding the file separately using **uudecode**. See "Sending large files via mail" (page 28) for further details.

To pipe a message through a filter:

1. Set the variable **cmd** to contain the name of the filter program to use.
2. Issue the | (pipe) command.

In this instance, we want to send the message through **uudecode**. To do this, type:

set cmd="uudecode"

When you type the **mail** command " | ", the contents of the current message will be decoded.

You do not see the decoded contents of the attachment. This is because **uudecode** automatically saves its output in a file with the name and permissions that appear at the top of the attachment. For more details about the format of encoded files, see "Sending large files via mail" (page 28).

You can read the decoded file using the command **!more** *filename* from the **mail** prompt (where *filename* is the name on the first line of the encoded attachment).

Sending large files via mail

Some mail sites and networks limit the size of message you can send. This limit may be as large as 100KB or as small as 10KB. If, for example, you want to include a large file (above 32KB) in a message, the recipient may find that part of the file has been lost during the transfer, although no error message will appear.

A solution to this problem is to compress and encode a file before including it in your message. Compressing a file reduces its size by about 60%; encoding the file increases the size of the compressed file by about 25%, so you end up with a compressed and encoded file that is about half the size of the original. If this still leaves a file that is too large to send, you should, if possible, split the file into smaller files before compressing and encoding them using the **split**(C) command.

Compressing and encoding a file

You should encode a compressed file before you send it because compressed files may include characters that **mail** cannot recognize. Although you may be able to send the compressed file, it may well hang the recipient's terminal when they try to read it.

Assume that you have a file called *myfile* that you want to send via mail to one of your colleagues. To compress the file, type:

compress myfile

The name of the file is changed to *myfile.Z*, where the "Z" identifies that the file is compressed. The original file is overwritten by the compressed file. (You can use the **-c** option with **compress** if you want to keep a copy of your original file.)

To encode the compressed file, type:

uuencode myfile.Z myfile.Z > myfile.Z.uue

(You can give the encoded file any name you wish, but it is sensible to attach *.uue* to the end of it to identify that it is a file encoded using **uuencode**.) You can then include *myfile.Z.uue* in a message as described in "Including a file or a message in mail" (page 20).

Decoding and uncompressing a file

A received message that includes a compressed and encoded file will look something like:

```
Hi Robert,

This is the bitmap image that I promised to send you
last week. I have compressed and uuencoded it.
Save this message, uudecode and uncompress it to
restore the image file:

begin 664 myfile.Z
M+DY4"DEF('EO=2!A<F4@:6YS=&%L;&EN9R!T:&4@3&EN:R!+:70@;VX@86X@
M7',M,4E"35QS*S$@36]D96P@-38O-3<Z"BAA*2!I;G-T86QL('1H92!,:6YK
    .
    .
    .
M(#4W+"!A<PID97-C<FEB960@8GD@=&AE(%QS+3%)0DU<<RLQ(#4W(%-U<'!L
896UE;G0@0V]V97(@3&5T=&5R+@HN3D0*

end

That's it - look forward to seeing the results!

------------------
See y'all soon. Vic
```

The first line of the encoded file included in the message reads:

```
begin 664 myfile.Z
```

The entry `myfile.Z` is the name that the decoded file will be given; `664` specifies the absolute octal permissions on the file.

Save the whole message as *a_message* (or give it any name you want).

To decode the file, type:

uudecode a_message

By default, the decoded file is called *myfile.Z*. If you wish to override this, use the **-s** option of **uudecode** and direct its standard output to the filename to be created; in this example, to *ourfile.Z*:

uudecode -s a_message > ourfile.Z

The final step is to uncompress the file. To uncompress *myfile.Z*, enter:

uncompress myfile.Z

The uncompressed file is called *myfile* (the name it had before it was compressed and encoded).

See also the **compress**(C) and **uuencode**(C) manual pages.

Running UNIX commands from within mail

You can execute a shell command from the **mail** prompt without leaving **mail** (or while editing a message). When the command finishes executing, you return to where you started the command. From the **mail**, precede the command with an exclamation point. For example, **!date** displays the current date without leaving **mail**.

While editing a message, you must precede the command with ~! at the start of a new line.

From the **mail** prompt, you can start a new shell with the command **sh** (short for **shell**). To exit from the new shell, press ⟨Ctrl⟩**D** or type **exit**.

Customizing mail

The mail service provides extensive facilities for customization. You can:

- have **mail** execute commands when it starts up
- divert incoming mail to specific folders
- reply automatically while you are on vacation
- notify you of incoming mail
- forward your messages to another person

Making mail execute commands at startup

When you start **mail**, it looks for a file called */usr/lib/mail/mailrc*. This optional file contains variables that customize **mail** for all users on the system.

After **mail** has executed the commands contained in */usr/lib/mail/mailrc*, it then looks for a file your home directory called *.mailrc*. If it finds this file, it executes the commands contained in it. You can set up *.mailrc* to customize your own mail environment. The commands in this file override the commands in */usr/lib/mail/mailrc*, but they only affect **mail** for you. A *.mailrc* might look like:

```
# set variables
# do not ask for the carbon copy list
set noaskcc
# always ask for the subject of a message
set asksubject
# list most recently received message first
set mchron
# if you are a recipient, do not delete
set metoo
# use pg(C) for paging messages
set crt=20
set PAGER=/usr/bin/pg
# allow mail to print its starting message
set noquiet
# put messages that have been read in mbox
set nohold

# personal aliases
# using domain addressing
alias andrea andrea@scribe.npr.com
# using UUCP addressing
alias dilly babage.dreadco.com!dilly
# using UUCP addressing via a path of several host machines
alias schroder ldsc!dsc!uunet!tomata.cs.ucsc.edu!schroder
# local aliases
alias XTRproject perry susant peterp kevinc
```

While using **mail**, you can get a list of the options currently set by typing **set** on its own. You can also set or unset individual variables, for example, to define make the screen scroll by 20 lines when you read a message, enter:

set crt=20

For full details of all the environmental variables available for **mail**, see the **mail**(C) manual page.

Automatic notification of new mail

You can configure your environment so that you are notified whenever new mail is sent to you, even if you are not in **mail**. To do this, set the **MAIL** shell variable to the pathname of your system mailbox. For example, if you are using the Bourne or Korn shells, you might set:

```
$ MAIL=/usr/spool/mail/login
$ export MAIL
```

You must set the **mail** shell variable if you are using the C shell:

```
$ setenv mail /usr/spool/mail/login
```

For details on how to set environment variables, see "Setting environment variables" in the *Operating System User's Guide*.

Redirecting incoming messages to other folders

Redirecting mail to folders other than your system mailbox is useful for keeping all the mail on a particular subject or the messages sent to an alias to which you belong in one folder.

To deliver mail to different folders, you must first set up the file *.maildelivery* in your home directory. To do this:

1. Create the *.maildelivery* file in your home directory:

   cd
   touch .maildelivery

2. Set the access permissions to 644 (`-rw-r--r--`):

   chmod 644 .maildelivery

3. Create a directory in which to put all the redirected mail. For example, create a directory called *myfolders* in your home directory:

   mkdir myfolders

   NOTE If you do not create the directory in which to redirect your mail before setting up *.maildelivery*, you can create an infinite loop because mail cannot be delivered to a nonexistent directory.

4. Lines in *.maildelivery* take the form of this example:

   ```
   #Header    Content    Action    Result    Filename
   to         login      >         ?         /usr/spool/mail/login
   ```

   The line beginning with the number sign "`#`" is a comment. Replace ***login*** with your login name.

   The "`>`" character in the "Action" column causes these messages to be added to (*/usr/spool/mail/**login***). The "`?`" in the "Result" column means that the message will be considered to have been delivered if it arrives safely in the system mailbox. If the message has already been delivered, the message will not be added to the mailbox. This is important if you do not want a message delivered to more than one folder.

   To redirect messages with specific patterns in the "Subject" header, specify **subject** in the "Header" column and the pattern to be matched in the "Content" column. For example, to redirect mail on the subject of "Style" to *myfolders/style.mail*, add the following line to *.maildelivery*:

   ```
   subject  "Style"    >    A    /u/login/myfolders/style.mail
   ```

5. Redirect mail addressed to any alias to which you belong. For example, if you want all messages sent to the *engr* alias to go to a folder called *myfolders/engr.mail*, add the following lines to *.maildelivery*:

```
to      engr    >       A    /u/login/myfolders/engr.mail
cc      engr    >       A    /u/login/myfolders/engr.mail
```

   The "A" in the "Result" column means that a message that has already been delivered will still be added to the folder.

   The *engr.mail* folder is created when it first receives mail; you do not need to create it yourself.

6. Add a line to deliver all the messages that are not caught by the conditions you have specified. Add the following lines to *.maildelivery*:

```
default   -         >    A     /usr/spool/mail/login
#
```

   The "default" in the "Header" column matches this field if the message has not already been delivered. The dash character (-) indicates that the "Content" field is not used here.

For more information on the format of the *.maildelivery* file, see the *maildelivery*(F) manual page.

Sending automatic responses when on vacation

You can use the **rcvtrip**(C) utility to generate automatic replies to messages while you are on vacation. Mail will still arrive in your system mailbox (or other folders, if you have set up */.maildelivery* to redirect messages).

rcvtrip only sends an automatic response to messages that include your login on the "To" or "Cc" lines. It cannot respond to messages sent to you as part of an alias. **rcvtrip** also logs those to whom it has responded, and sends only one reply to each person.

Use the following procedure to set up **rcvtrip**:

1. Create a file called *tripnote* in your home directory to contain the response that you want sent. For example, you might want to tell people about where you are and when you will be back.
2. Make sure the permissions on *tripnote* are set to 644 (`-rw-r--r--`):

   chmod 644 tripnote

3. Create an empty file called *triplog* in your home directory:

   touch triplog

   On your return, *triplog* will contain a record of the people responded to by **rcvtrp**. Each line in *triplog* lists the address, date, and time you received the message, part of the "Subject" line, and a plus "+" or minus "-" character to indicate whether or not **rcvtrip** sent an automatic response.

4. Make sure that the permissions on *triplog* are set to 644:

   chmod 644 triplog

5. Add the following as the first line in your *.maildelivery* file:

   ```
   *  -  pipe  R  rcvtrip $(sender)
   ```

   This makes **rcvtrip** process all your incoming messages before they are processed according to the remaining instructions in *.maildelivery*, and sent to the appropriate folders. For more details, see the **rcvtrip**(C) manual page.

   You should leave this line commented out (using a leading number sign "#") until immediately before you go on vacation.

6. On your return, comment out the **rcvtrip** line in *.maildelivery*.

Forwarding your messages automatically

The **resend**(C) utility allows you to forward all your incoming messages to another person automatically. For example, you might want someone else to read your mail when you go on vacation to ensure that any urgent mail is answered promptly. As with **rcvtrip**, you use **resend** in your *.maildelivery* file. Use the following steps to set up **resend**:

1. If you have not already done so, create a file called *.maildelivery* in your home directory and make sure its permissions are set to 644. For details on how to do this, see "Redirecting incoming messages to other folders" (page 32).

2. Add the following first line to *.maildelivery*:

   ```
   To  login  |  A  /usr/bin/resend  user_2
   ```

   Replace ***login*** with your login name and ***user_2*** with the address of the person to be forwarded your messages.

For more information on the format of the *.maildelivery* file, see the *maildelivery*(F) manual page or "Redirecting incoming messages to other folders" (page 32). For more information on the **resend** utility, see the **resend**(C) manual page.

More about mail

The collection, sorting, transport, and delivery of e-mail over a network is handled by two utilities:

MUA The Mail User Agent is the program that you use to create, send, read, and organize your messages. SCO systems provide the choice of several MUAs: **mail**, **SCO Shell Mail** (page 39), and **scomail**(XC).

MTA The Mail Transport Agent is the program that transports mail, puts the messages you send in the outgoing mail queue, and reads messages from the incoming queue to deliver to your mailbox.

A choice of two MTAs is provided with SCO systems. These are MMDF (Multichannel Memorandum Distribution Facility), described more fully in Chapter 4, "Managing mail with MMDF" (page 67), and **sendmail**, described in Chapter 5, "sendmail administration" (page 149).

Other ways of contacting users

To list the users currently logged in to your system, use the **who** command. The login, terminal name, and login date and time are displayed:

```
$ who
josephh     tty001       Mar 25 14:29
susant      tty002       Mar 25 14:29
peterp      tty003       Mar 25 14:29
perry       tty005       Mar 25 14:31
```

To find out more about an individual user, use the **finger** command. For example:

```
$ finger perry
Login name: perry                In real life: Perry Patetic
Directory: /u/perry              Shell: /bin/ksh
On since Mar  9 09:23:40 on tty005
No Plan.
$
```

finger can be used on a user who is not logged in. If you do not specify a login, **finger** displays information about all the users currently logged in.

Note that in the example above, the bottom line states `No Plan`. This indicates that the user does not have a file called *.plan* in their home directory. If such a file exists, its contents are printed by **finger**. The *.plan* file is a useful place to put information that you want to have publicly available, such as your telephone number and current work project. Use any text editor to create a *.plan* file.

See also the **who**(C) and **finger**(C) manual pages.

Holding conversations using talk

You can write messages to users currently logged in to your system by using the **talk** command. This is useful when you need to communicate with someone at another terminal immediately.

To establish a connection using **talk**, use the command:

talk ***login***

where ***login*** is the login name of the person that you want to talk to.

A message appears on their terminal, asking them to "talk" back to you. Once they run **talk** (using your login as the argument), your terminal displays two windows; one shows anything that you type, the other shows anything that they type:

For example, assuming your login is *perry*, and you want to talk to *nigella*, type:

talk nigella

Nigella's terminal beeps, and a message appears:

```
Message from Talk_Daemon@machina.plains.sit.COM at 9:49 ...
talk: connection requested by perry@machina.plains.sit.COM
talk: respond with: talk perry@machina.plains.sit.COM
```

To reply, *nigella* types:

talk perry

Everything *perry* types appears in the upper window on his screen, and in the lower window on *nigella*'s screen. The **talk** conversation occurs in real time with each character being sent as you type it. This means that the other person will be able to watch you correct your spelling mistakes, and may even comment on how slowly you type!

To quit **talk**, press ⟨Del⟩ (or your interrupt character).

NOTE If a user is logged in with multiscreens or on different terminals, you will need to specify the terminal to **talk** to. Use the command **w** to list users and terminals. It is a bad idea to **talk** to a terminal running an interactive application like **mail**, **vi** or the SCO Shell, because this may corrupt the screen display.

A similar facility to **talk** is provided by **write**(C). This sends a line at a time to the other person and it does not use a special split-screen display.

Preventing someone from writing a message to you

To prevent messages appearing on your screen while you are working, you can use the **mesg** command, which lets you accept or refuse messages sent to you via **write**. Type:

mesg n

to prevent messages from being sent to your terminal, and:

mesg y

to accept them.

To find out your current status ("y" or "n"), type **mesg** on its own. To find out whether other users will accept messages sent via **write**, type **finger**. The display will be something like:

```
Login    Name                TTY  Idle    When          Office
josephh  Joseph Hulot        001          Mon 14:29
susant   Susan Tzarcha      *002     3    Mon 14:29
peterp   Peter Piper         003          Mon 14:29
perry    Perry Patetic      *005          Mon 14:31
```

The asterisk (*) next to the TTY number indicates that the user is not accepting messages. Messages sent by *root* ignore the **mesg n** setting.

If you are using **mail** or **vi** and a message arrives that corrupts your screen, you can type ⟨Ctrl⟩**L** to redraw the screen. See also the **finger**(C) and **mesg**(C) manual pages.

Chapter 3

Using SCO Shell Mail

SCO Shell Mail allows you to compose, send, receive, forward, and reply to interoffice messages from your computer terminal. It can be a complete document-routing and distribution system for your organization. The **scosh**(C) mail reader is also available on the SCO OpenServer Desktop System.

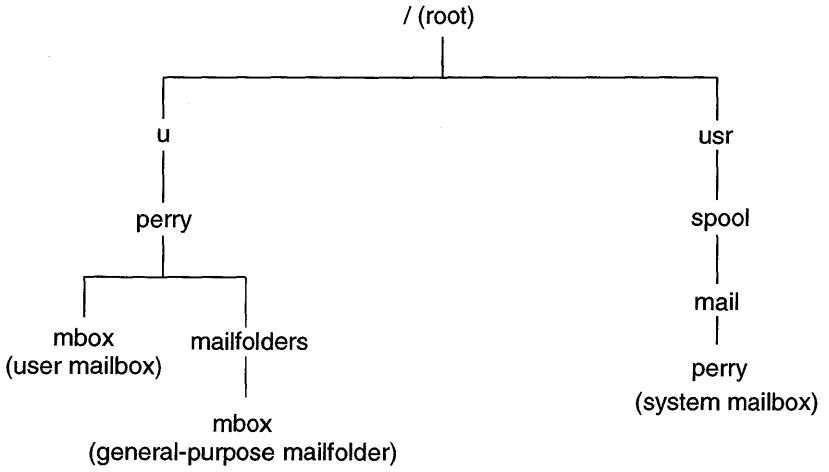
You can exchange mail with any of the users on your computer. If your computer is linked to others via a network, you can also exchange mail with users on other machines.

When you schedule events through the SCO Shell Calendar application, notices can be sent automatically through e-mail to everyone you invited.

It is useful to think of the e-mail system as a postal system, with the post office being represented by the system mailbox. The system mailbox is located in the */usr/spool/mail* directory, and contains a file for each user on your system. Your own personal or user mailbox is the file named *mbox* in your home directory. Mail sent to you is put in your system mailbox and is automatically saved in your user mailbox after you have read it.

When you run the mail program from SCO Shell, you also have a default mail folder in the *mailfolders* directory in your home directory. This is a general purpose file where you can save messages. For further details on using mail folders, see "Organizing your mail" (page 49).

For user *perry*, the default names and locations of the files mentioned above are shown below:

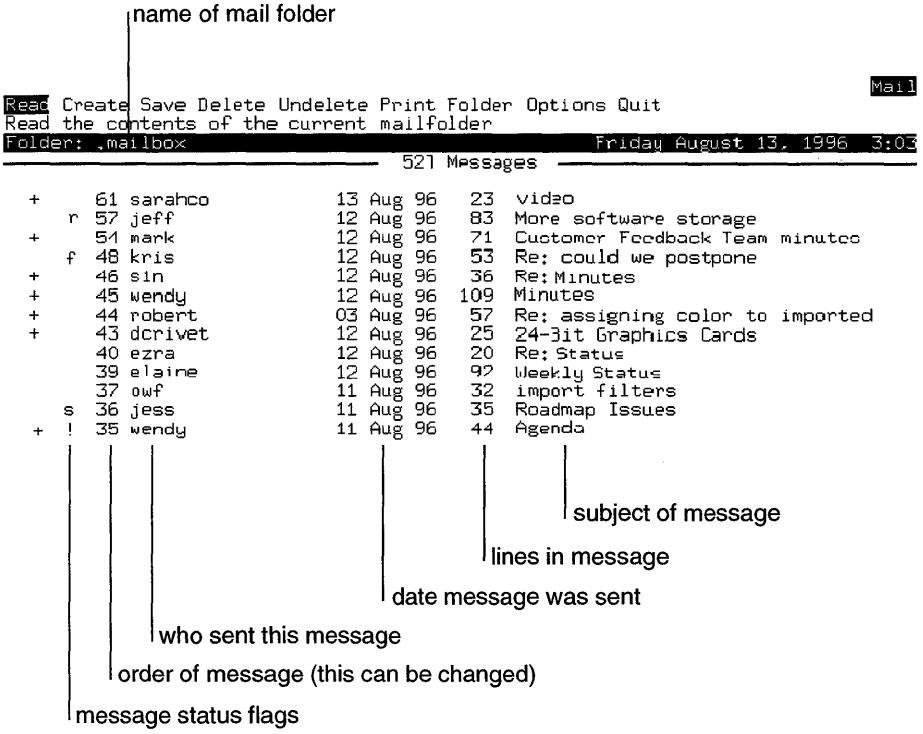


For further information on default mail filenames and locations, see the **mailx**(C) manual page.

Starting mail

Start **SCO Shell Mail** either by entering **scosh email** at the command line or by starting SCO Shell and selecting the Mail Application. For further instructions on SCO Shell, see "Starting SCO Shell" in the *Operating System User's Guide*.

The mail opening screen shows information about messages waiting for you in your system mailbox. New messages are added to the end of the message list as they arrive:



Message status symbols (called flags) can appear in the left margin of the message list next to the individual listings. These characters tell you facts about the messages: whether or not you have read this message, whether it has an attachment, whether you have replied to it, and so on. The following table shows a complete list of message status flags.

Symbol	Description
+	A message that has not been read yet. The "+" disappears only after you read the message.
a	A message that has attachments.
!	An urgent message that requires immediate attention.
s	You have saved a copy of the message to a file or mail folder.
f	You have forwarded a copy of the message to others by selecting **Forward** from the **Read** menu.
r	You have sent a reply to the sender by selecting **Reply** from the **Read** menu.

Further details on these flags are given at appropriate places in this chapter.

NOTE In SCO Shell Mail, the date displayed in the message list is incorrectly formatted for the German locale. The last digit of the year is missing from the display.

Quitting from mail

Select **Quit** from the **Mail** menu. Then select **Yes** to save the changes you have made to the message list, **No** to return to mail, or **Abandon** to leave mail without saving any changes you have made to the message list.

Abandon does not undo any mail that you have sent out, including forwarded mail and replies. It also does not affect mail that you have saved to a file or mail folder (page 45).

Getting help when you are using e-mail

To get online help when you are using mail, press ⟨F1⟩ at any time. An initial help screen is displayed giving general help about the part of **SCO Shell Mail** you were using when you pressed ⟨F1⟩. With most help screens, you can press ⟨F1⟩ again to get more detailed help, or press ⟨Esc⟩ to get out of help.

Reading mail

When you are sent a message, it is put in your system mailbox file. When you start SCO Shell Mail, your system mailbox is loaded. You then see a list of all messages in your mailbox and can read any of them.

To read your mail messages, select **Read** from the **Mail** menu. Move the highlight to the message you want to read and press ⟨Enter⟩. The message you selected appears below the message header ("To", "From", and "Subject"). If the message has an attachment, the header information tells you so.

Once you begin reading the message, you can select **Continue** from the **Read** menu or use the cursor keys on your terminal to move forwards and backwards through it. For a full list of keyboard commands, select **Help** from the **Read** menu, or press ⟨F1⟩.

You can redisplay the message headers after you have begun to read your messages by selecting **Quit** from the **Read** menu.

Speed-reading mail

To quickly read through several messages, select **Read** from the **Mail** menu. Subsequent messages are displayed in turn as you press ⟨Enter⟩.

Reading an attachment

SCO Shell Mail allows you to send a separate file or clipboard item along with mail messages. Unlike the message itself, this attachment can be a file with a special data format, such as a spreadsheet file, word processing document or binary program.

An "a" next to a message in the header list tells you that the message has an attachment. The message header tells you the name of the attached file ("File Attached") and its file type ("Attachment Type"). If the attached file has a special file format, it is automatically translated for you.

To view the attachment, you must first display the message (containing the attachment) on the screen, then select:

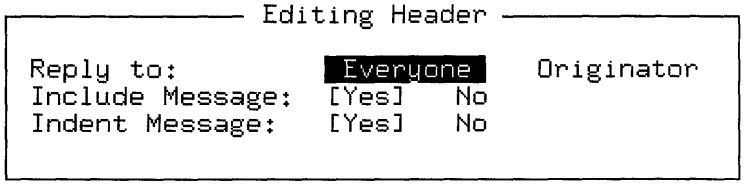
Extra ⇨ Attachment

The attachment appears on the screen, and you can scroll through it with the cursor movement keys. You can save the attachment to a file; see "Saving a message with an attachment" (page 47) for details.

If **SCO Shell Mail** cannot translate the attachment's file format, an error message appears. Press ⟨Enter⟩ to return to the **Read** menu and save the attachment to a file. You should then seek advice on decoding the file from your system administrator.

Replying to a message

To reply to the message you are currently reading, select **Reply**.



You can specify who you want your reply to be sent to, whether to include the text of the original message in your reply, and whether to indent the text of the original message from the rest of your reply.

When you finish filling in the **Editing header** form, press ⟨Enter⟩ in the last field, or ⟨Ctrl⟩**X** to enter edit mode. If you chose to include the original message, the text of the message is displayed.

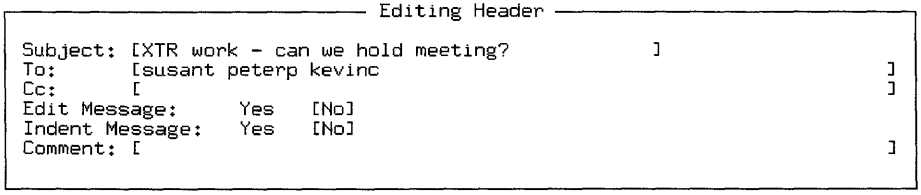
Compose your reply, then send it by pressing ⟨Esc⟩ and selecting **Deliver**. The Read screen then reappears, with the original message still displayed.

The next time you see a list of your messages, the message replied to will be marked with an "r".

Note that the **Reply** form does not let you change the message's subject, or send the reply to anyone besides the originator and the original recipients. You can change the details in the message header while you are editing your reply by selecting **Header**. See "Editing a message's address before sending the message" (page 54).

Forwarding a message to another user

To forward the message you are currently reading, select **Forward**.



You can add a brief comment to the message or edit the message before you send it. You can also change the subject of the message.

You can change the message using the same editing commands that you use for composing new messages. When you finish, forward the message by selecting **Deliver** from the **Create** menu. You then return to the Read screen, with the original message still on display. Your changes affect only the copy that you forwarded, not the original message.

In the "Comment" field, you can enter a one-line comment to go with the message. **SCO Shell Mail** inserts the comment, along with your login, at the top of the message.

If the message has an attachment, it is forwarded along with the message.

When you next see the message list, the message you forwarded is marked with an "f".

Changing the comment line in forwarded messages

When you forward a message, you can add a brief comment to the message without editing it. The recipient of the message sees the comment at the top of the message. If you want, you can change this standard introduction line to something else for all messages that you forward with a comment. Select:

Options ⇨ Customize

and enter the introduction that you prefer in the "Forward Comment Introducer" field.

Saving a message

You can save one or more messages to a file or mail folder for permanent storage. If you are saving a message with attachments, you can save the message, the message, or both. For more information about mail folders, see "Organizing your mail" (page 49).

After you save a message, it normally stays on the message list until you delete it. An "s" appears by its listing to remind you that you saved it. If you select **Preferences** from the **Options** menu, and then select "Delete Saved", saved messages are automatically deleted from your system mailbox and thus no longer appear on the message list. See "Automatically deleting saved messages" (page 48).

To save a message, select **Save** from the **Mail** menu. Select the messages to save from the list displayed, and press ⟨Enter⟩.

You can also save messages by entering the message number or numbers. Enter several numbers by separating them with spaces (for example, "1 3 7"), a range of numbers (for example, "1-5"), or a combination of both (for example, "1-5 8 11"). Press ⟨Enter⟩ to save the specified messages.

Having selected one or more messages to save, use the "Save To" field to indicate whether you want to save the messages to a file or mail folder.

Saving a message to a file

To save the messages to a file, select **File** from the "Save To" field. Enter a filename in the "Name" field. You may either enter an existing filename to send the messages to an existing file, or enter a new filename to save the messages to a new file. If you want to save them to a file in a different directory, you must include the file's pathname.

If you chose to save the messages to an existing file, select **Append** to add the messages to the end of the file, or **Overwrite** if you want the messages to replace the current contents of the file.

Saving a message to a mail folder

If you wish to save messages to a mail folder, select **Mail-Folder** from the "Save To" field. When you do this, the default folder name, *mbox*, is displayed in the "Name" field. To save the messages to *mbox*, press ⟨Enter⟩. Otherwise, enter the name of the mail folder that you want. If that folder is inside a drawer (a subdirectory) in the *mailfolders* directory, enter the name of the drawer followed by a slash and the name of the folder. For example, enter **Sales/Prospects** for a folder named *Prospects* in a drawer named *Sales*. For more information on mail folders and drawers, see "Organizing your mail" (page 49).

If you enter the name of an existing mail folder, the messages are appended to the end of that folder. If you enter a folder name that does not exist, or enter the folder name along with a drawer name that does not exist, you are asked whether you want to create a new mail folder. If you select **Yes**, **SCO Shell Mail** creates a new mail folder with the name you supplied and saves the message to it. If you try to save the message to a folder that is inside a drawer does not exist in the *mailfolders* directory, selecting **Yes** will create both the new drawer and a new folder inside that drawer.

The header information for each message that is saved to a mail folder is included by default. **SCO Shell Mail** can read any mail folder containing messages that include the header information.

Automatically saving a message when you have read it

You can save messages automatically once you have read them. To do this, select **Options ⇨ Preferences** from the **Mail** menu. Move the highlight to the "Auto Mbox" field and press ⟨Space⟩ to select it. Now all messages that you read are saved to your *mbox* mail folder automatically when you exit **SCO Shell Mail**, unless you delete them or save them to another file folder before exiting.

Saving a message with an attachment

If the mail message that you are currently saving has an attachment, an additional form is displayed after you have responded to the **Saving Messages** form. The form shows you the header listing for the message that has the attachment, plus the attachment's original filename. To save the attachment under its original filename, press ⟨Enter⟩. To save it with a different filename, enter the new name and press ⟨Enter⟩. For example, if the attachment is a file that you are going to use with an application, make sure that you give it a filename that the application recognizes. In either case, **SCO Shell Mail** saves the attachment as a file in your current directory and returns you to the **Mail** menu. You can save an attachment only as an ordinary file, not as a mail folder. Saving the attachment does not delete it. The attachment remains with the message until the message is deleted or saved to a file.

Deleting a message

Deleted messages are not removed from your system mailbox completely until you exit mail (page 42) or switch to a new mail folder (page 53).

To remove one or more messages from the message list, select **Delete** from the **Mail** menu. Select one or more messages using the ⟨Space⟩ bar and arrow keys; press ⟨Enter⟩ to delete these messages.

To update the message list so the deleted messages are removed and the remaining messages are numbered consecutively, select:

Options ➪ Switch ➪ System

Restoring a message

You can only restore messages that you have deleted during your current mail session. Also, if you switch to a new mail folder (page 53), any messages you deleted are permanently gone.

To restore a message(s), select **Undelete**. Mark the message(s) to be restored, and press ⟨Enter⟩.

You can also restore a message by entering its message number after selecting **Undelete**. Enter several numbers by separating them with spaces (for example, "1 3 7"), a range of numbers (for example, "1-5"), or a combination of both (for example, "1-5 8 11"). Press ⟨Enter⟩ to restore these messages.

After you choose the messages you want, you automatically return to the listing of all the mail messages. The restored message(s) are now displayed.

If you press ⟨F8⟩ while reading a message, **SCO Shell Mail** brings back the last message that you deleted. The message appears on the screen immediately.

Automatically deleting saved messages

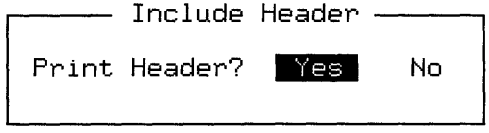
To delete messages automatically from the system mailbox when you save them to a file or mail folder, select **Preferences** from the **Options** menu, and then select "Delete Saved". Once you do this, messages disappear from the message list as soon as you save them. You can restore a message to the message list (page 47) by selecting **Undelete**.

Printing a message

You have two options for printing messages: you can print the message you are currently reading, or you can select and print one or more messages from the message list.

To print the message you are currently reading, select:

Print ➪ Go



Select **Yes** to include the header information or **No** to print the message only. The message is sent to the queue of the printer you are currently using.

To print several messages, you must first return to the **Mail** menu. Select:

Print ➪ Go

Select the message(s) to be printed, and press ⟨Enter⟩. Then select **Yes** to include the header information or **No** to print the message only. The message is sent to the queue of the printer you are currently using.

If you want to send messages to a different printer, select:

Print ➪ Select

A list shows all the printers that are available to you. A message at the top of the screen tells you what printer you are currently using. To switch to a different one, highlight the printer you want and press ⟨Enter⟩.

Your new printer selection stays in force only until you leave **SCO Shell Mail**. Then it changes back to the default printer selection for SCO Shell. To change your default printer, select:

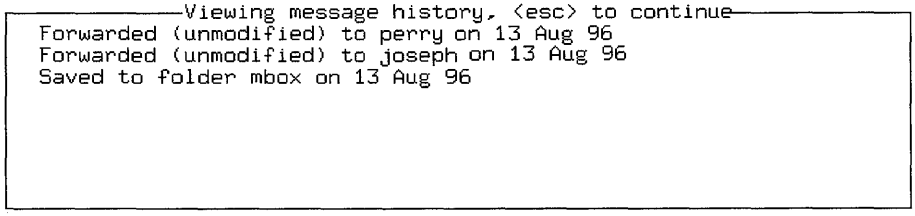
Print ➪ Select

from the main SCO Shell menu.

Viewing the history of a message

Once a message reaches your system mailbox, **SCO Shell Mail** keeps a history of what you do with it. It takes note whenever you save the message or its attachment, forward the message, or reply to the message. You might want to examine the history of a message to see if you replied to a particular person, or when you replied. To see the history of the current message, select:

Extra ⇨ History



The message history includes a separate line for each operation you performed with this message, including the date of the operation. When you delete a message from the system mailbox or save a message to a file, its history is lost, but if you save the message to a mail folder, the message's history is also saved there. When you load a mail folder into **SCO Shell Mail** and read it, selecting **History** shows you a message's history, if any, up to the time when you saved the message from the system mailbox.

Organizing your mail

Mail can be saved to files called mail folders. You can create as many as you want, and with any name you choose.

As you read the mail in your system mailbox, you can decide which messages you want to keep, and save them to the appropriate mail folders.

When you want to read the contents of a mail folder, you can look at the contents of that folder instead of your system mailbox file. Listings for the messages in the mail folder then appear on your screen, and you can choose the messages that you want to look at. You can also delete messages from the mail folder, forward copies of old messages to others, and do anything else you would do with new mail from the system mailbox.

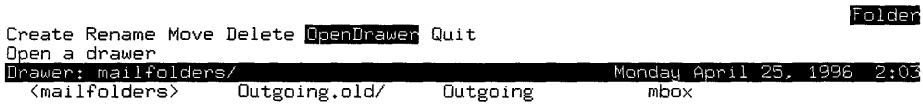
Mail folders are located in the *mailfolders* directory, which is a directory in your home directory.

SCO Shell Mail lists all of your mail folders and lets you choose the one you want.

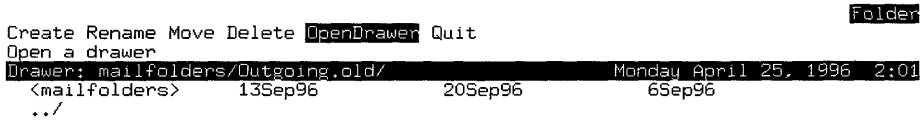
If you create a lot of mail folders, you may eventually begin to lose track of them. To stay organized, you can create subdirectories or drawers within the *mailfolders* directory and distribute your folders among them. You can create a drawer to hold certain types of mail folder (for personal correspondence, for example, where each folder in the drawer holds mail from one of the people that you correspond with). You can also create a drawer for mail about customers, with each folder in the drawer holding mail about a specific customer.

Viewing the contents of a folder or drawer

To examine the contents of one of the drawers within the *mailfolders* directory, select **Folder**. The point-and-pick list now shows drawers only:



Select **OpenDrawer**, and pick the drawer you want to see, (for example *Outgoing.old*); its contents appear on the screen and the divider line displays its name:



Each drawer under the *mailfolders* directory always contains two standard entries: *<mailfolders>* and *../*. Select *<mailfolders>* to return to the main *mailfolders* directory. You can do this from within any drawer, even if that drawer is inside another drawer (that is, a subdirectory of a subdirectory). Select *../* to return to the drawer above the drawer that you are in.

Starting mail from a different mail folder

To organize your mail, you may wish mail from certain users to be redirected automatically to other mail folders than the default mail folder. Do this by following the procedure described in "Redirecting incoming messages to other folders" (page 32).

To start **SCO Shell Mail** from a different mail folder from the default system folder, select:

Options ➪ Applist ➪ Edit

from the main SCO Shell menu, choose the entry for Mail from the application list.

Add the option **-f** to the end of the path shown in the "Path Name" field, followed by the path leading to the mail folder you want. For example, if your login is *nigel*, your home directory is */u/nigel*, and you want to load your general purpose mail folder, you would change the "Path Name" field to read:

$OALIB/email -f /usr/nigel/mailfolders/mbox

If you choose an alternate startup mail folder, you can still read new mail in your system mailbox. Select:

Option ➪ Switch ➪ System

to load the system mail folder into **SCO Shell Mail**.

Creating a new folder or drawer

To create a new folder or drawer, select:

Folder ➪ Create ➪ Folder

or

Folder ➪ Create ➪ Drawer

SCO Shell Mail prompts you for the name of the folder or drawer to be created. Enter the name, and press ⟨Enter⟩.

By default, the folder or drawer is created in whatever drawer is currently showing on the screen. If you want to put the new folder or drawer in a different drawer, select **OpenDrawer** first to bring that drawer to the screen.

You can also create drawers and folders while saving a mail message, by selecting **Save** from the **Mail** or **Read** menus. See "Saving a message to a mail folder" (page 46).

You can create a folder inside a different drawer than the one that is currently open. You do this by entering the pathname to the drawer where you want the new folder to be. You must first create the drawer if it is does not already exist.

Suppose you are in the *mailfolders* directory, which contains a drawer (or subdirectory) called *projects*. To create a folder named *proposals* inside the *projects* drawer, you would enter the path *projects/proposals*. The new folder then appears inside the *projects* drawer, even though *mailfolders* is still on the screen.

Select **Drawer** to create a new drawer. **SCO Shell Mail** prompts you to enter the name of the drawer. Enter it and press ⟨Enter⟩.

By default, **SCO Shell Mail** puts the new drawer inside whatever drawer you have displayed on the screen.

If you want to create a drawer inside a different drawer than the one that is currently open, you have two options. First, you can select **OpenDrawer** to open the drawer, and then select **Create** again to create the drawer. Second, you can enter the directory path from the current drawer to the drawer where you want the new drawer to be created.

Renaming a folder or drawer

To rename a folder or drawer, select:

Folder ➪ Rename

Enter the name of the folder or drawer that you want to rename.

If the folder or drawer that you want to rename is not in the current drawer, you can use the ">" key to display the contents of other drawers without actually selecting them for renaming. When you highlight a drawer and press ⟨>⟩, the contents of that drawer appear on the screen. You can then select one of the drawers or folders in the new drawer. To display the drawer above the currently displayed drawer, press ⟨>⟩ on the ../ entry. To display the *mailfolders* directory, if that directory is not on display, press ⟨>⟩ on the *<mailfolders>* entry.

Enter the new name for the folder or drawer and press ⟨Enter⟩.

SCO Shell Mail renames the folder or drawer, and the **Folder** menu reappears.

Moving a folder

You might want to move a folder between drawers in order to reorganize your mail folders. To move a folder from one drawer to another, select:

Folder ➪ Move

You cannot move a drawer, only folders.

Highlight the folder(s) that you want to move. Highlight the drawer that you want to move the folders to, and press ⟨Enter⟩. The folders move from their current location to the new folder.

Deleting a folder or a drawer

To delete a folder or drawer, select:

Folder ➪ Delete

Highlight the folder(s) or drawer(s) to be deleted, and press ⟨Enter⟩.

Switching to a new mail folder

By default, **SCO Shell Mail** displays the new messages in your system mailbox file, not the old messages that you have saved in mail folders.

To look at the messages in one of your folders instead of in the system mailbox, select:

Options ➪ Switch

Then select the mail folder you wish to examine.

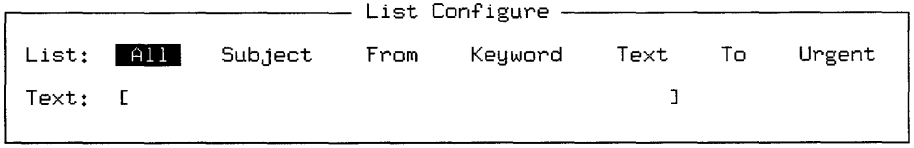
Listing specified mail messages

SCO Shell Mail displays all your messages by default.

To list only messages that match a specific condition, select:

Options ➪ List

You are then prompted to choose from among a set of options; whether to list messages on the basis of subject, who originated them, keyword, title, some text, their destination, or whether they are tagged as urgent:



Enter the information to search for and press ⟨Enter⟩; **SCO Shell Mail** shows you only those messages that match.

To go back to viewing of all your messages, select **All**.

Changing the format of the message list

To change the format of the message header list, select **Preferences** from the **Options** menu. Experiment to see which is most suitable for you.

To display the most recently displayed message first, select "Reverse List". If "Reverse List" is not selected, the most recently received message is displayed last.

To save the new format as the default, select **Yes** in the "Save as default" field. This tells **SCO Shell Mail** to retain these new preferences as the default, even after you end this session.

Creating and sending a message

You can send e-mail to anyone who uses your SCO system. If your system communicates with a computer network, you can send mail to other users who are linked to your computer network.

Once you have sent mail, there is no way to recall it, so be careful. It is a good idea to apply the same standards to e-mail that you would apply to written letters. A good precaution is to use **SCO Shell Mail**'s built-in delay facility (page 57), which waits for a certain time period before dispatching mail messages.

To create and send a new message, select **Create** from the **Mail** menu. On the blank mail header form that appears, fill in the address for your mail:

- the subject of the message
- the logins of the people you are sending it to
- the names of the people who should receive carbon copies

Press ⟨F3⟩ in the "To" or "Cc" fields to see a window that lets you search through lists of aliases (page 60).

When finished, press ⟨Ctrl⟩**X** to begin creating your message.

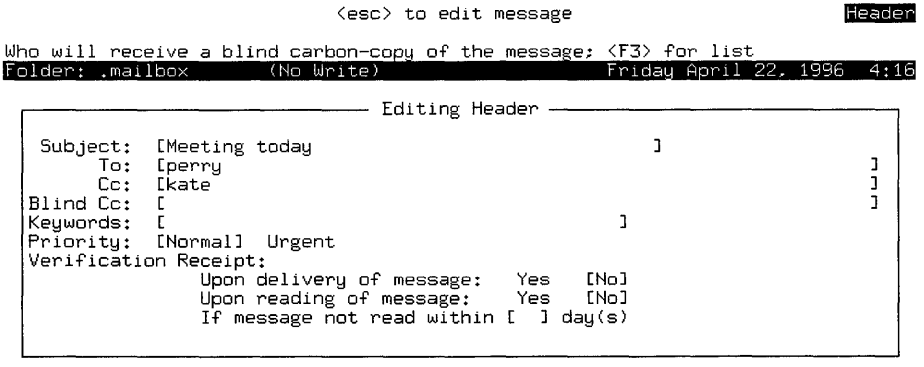
Type your message onto the screen. You do not have to type ⟨Enter⟩ at the end of each line, only when you want to start a new paragraph. You can use the editing keys listed in Appendix A, "mail(C) commands" (page 233) to correct mistakes. See also "Editing a message before you send it" (page 55).

To adjust the column width of messages you create, select **Preferences** from the **Options** menu, and enter a number in the "Line Width" field. The default line width is 65 characters.

When you are finished, press ⟨Esc⟩ and select **Deliver** to send the message.

Editing a message's address before sending the message

While you are composing your message, you may edit or correct the message's header by selecting **Header**. A form is displayed that displays the information you entered on the header in the first three fields. You can change this information if you want.



Note that this form also contains several fields that are not on the original header form because they offer optional header features that you may not always need.

In the "Blind Cc" field, enter the logins of the users who you wish to receive a copy of your message, but whose login(s) you do not want displayed to the other users on the distribution list. You might want to do this to keep recipients from knowing that others have received the same information.

The "Priority" field indicates whether the message should have normal priority or urgent priority. On the message list, an exclamation mark (!) appears next to each urgent message as a signal that it should be read at once.

The "Verification Receipt" field can be used to send you an automatic reply when the message is received or read. It can also be used to notify you if the message has not been read after a given number of days.

Editing a message before you send it

To edit a message you are creating, press ⟨Esc⟩ and select **Edit** from the **Create** menu to start the editor program configured for your account. Exit the editor using its normal quit command.

You can edit a message using a text processing program other than the editor built into **SCO Shell Mail**.

To define a different standard editor, select:

Options ⇨ Preferences

from the **Mail** menu.

```
                    Preferences

Auto Mbox:       [*]    Reverse List:    [*]
Include Self:    [*]    Save Outgoing:   [*]
Auto Editor:     [ ]    Prompt for Cc:   [*]
Edit at Top:     [*]    Delete Saved:    [ ]
Verify Addresses before Delivery:        [*]

Line Width:      [67]
Delay Delivery: [15] minutes
Archive Outgoing folder every [7  ] days

Mail Editor:     [                                        ]

Header list format: [Standard]   XENIX    Two-Column

Save as default:     Yes    No
```

Move the highlight to the Mail Editor line, and change the pathname for the editor program to the one you want to use. Ask your system administrator if you do not know the pathname of the editor program.

Canceling a message before you send it

If you decide that you do not want to continue with the message you are currently creating, you can cancel it. When you do this, any text you have entered is discarded.

While you are creating a message, press ⟨Esc⟩ and select **Quit**. You are prompted to hold the message:

```
                    Hold Message

Hold this message?   Yes    No
Hold Name:  [                                          ]
```

Press ⟨Enter⟩ to select the default **No**.

Holding a message for delivery at a later time

You can hold a message for further editing or delivery at a later time.

Press ⟨Esc⟩ while you are creating a message, and select **Quit**. When prompted to hold the message, select **Yes**, and give a hold name; this helps you identify this message among several held messages.

Retrieving a held message

If you have sent messages to a hold file, you are reminded of this each time you select **Create** from the **Mail** menu. If you want to retrieve a message that you have held, select **Yes** in the "Retrieve a held message?" field. Enter the name of the file you want to retrieve. By default, the form displays the name of the hold file you created most recently. If that is the file you want, press ⟨Enter⟩.

If you do not remember what name to enter, press ⟨F3⟩ to display a list of all messages that you saved to hold files; highlight the name of the one that you want and press ⟨Enter⟩.

The message you selected is retrieved from the hold file and displayed on the screen. When you have retrieved a file, **SCO Shell Mail** automatically places you in **Edit** mode.

When you retrieve a held message, it is no longer held. If you want to stop work and save the message again, you must hold it again on quitting.

Delaying the delivery of a message

You may have second thoughts about sending a mail message after you have sent it. To give yourself time to cancel messages, select:

Options ➪ Preferences

and enter the desired delay in the "Delay Delivery" field. This delays the delivery of all messages that you create by that number of minutes. The default entry is zero, meaning that there is no delay. The delay period starts when you complete a message and select **Deliver**.

Retrieving a delayed message

During the delay period, you can edit the message or cancel it by selecting **Outbox** from the **Options** menu. You can also send the message immediately instead of waiting for the delay period to expire.

To inspect your delayed messages, select **Outbox** from the **Options** menu. The screen shows the messages that are currently in your *Outbox*. If you leave them alone, **SCO Shell Mail** delivers each of them when its delivery delay period is over.

Editing a delayed message

Select **Retrieve** from the **Outbox** menu to edit one of the messages in the *Outbox* before it is delivered. **SCO Shell Mail** now highlights the first message on the *Outbox* list. Select the one you want.

SCO Shell Mail now puts you in Edit mode, and you can make changes to the message. You have access to all editing keyboard commands and the **Create** menu. When you finish, select **Deliver** from the **Create** menu. The edited message returns to the *Outbox* for the remainder of its waiting period. The **Mail** menu and message list reappear on the screen.

Sending all delayed messages now

You may sometimes want to send a piece of mail out right away instead of letting it wait in the *Outbox*. To have **SCO Shell Mail** deliver all messages currently waiting in the *Outbox*, select **Flush**.

When you leave **SCO Shell Mail**, it automatically delivers any messages still waiting in the *Outbox*.

Canceling a delayed message

Select **Cancel** from the **Outbox** menu to cancel a message that is waiting in the *Outbox*. **SCO Shell Mail** deletes the message instead of delivering it.

When you select **Cancel**, **SCO Shell Mail** highlights the first message on the *Outbox* list. Use the arrow keys to move the highlight to the message you want, and then press ⟨Enter⟩. The message you chose is canceled, and it disappears from the *Outbox* list. You then return to the **Mail** menu.

Saving the messages you send

You can save a copy of all messages you send, including replies and forwards. To do this, select **Preferences** from the **Options** menu. Move the highlight to the "Save Outgoing" field, and press the ⟨Space⟩ bar to turn on the feature. An asterisk indicates the feature is turned on.

The copies of outgoing mail are saved in a mail folder named *Outgoing*. You can access this folder by selecting **Switch** from the **Options** menu.

If you send a lot of mail, this folder can become quite large. You may want to archive the folder (this page) periodically and start again.

Archiving your outgoing mail

To automatically move the contents of your *Outgoing* mail folder to archive files at regular intervals, select **Preferences** from the **Options** menu, and then select "Archive Outgoing Folder". Enter the interval that you want, in days.

This field is only effective if you have selected "Save Outgoing" (this page) from the **Preferences** form, which automatically saves all your outgoing correspondence to the *Outgoing* folder. If you do not empty this folder at intervals, it grows very large.

The archiving process moves the *Outgoing* folder to a subdirectory of the *mailfolders* directory called *Outgoing.old*. There, the folder is renamed according to the current date. For example, *Outgoing.old* might contain folders named *6Sep94*, *13Sep94*, and *20Sep94*, if you chose to archive your *Outgoing* folder every seven days. Each folder holds the contents of the *Outgoing* folder as of the date in the folder name.

After the archive process takes place, there is temporarily no *Outgoing* folder in your *mailfolders* directory. **SCO Shell Mail** creates a new one as soon as you send your next message.

Including a file or a message in mail

You can include a text file or another mail message in mail that you create.

To include a file in a message you are creating, press ⟨Esc⟩ and select **Include**, then select **File** from the "Include" field. The cursor skips directly to the "Name" field. Enter the name of the file. If it is not in your current directory, enter the file's full pathname. You can also press ⟨F3⟩ for a point-and-pick list of files.

To include a mail message, select **Message** from the "Include" field. Enter the message's number in the "Message Number" field. If you do not know the number, press ⟨F3⟩ for a point-and-pick list of all the messages in the system mailbox or current mail folder. In the "Indent Message" field, select **Yes** if you want the included message to be indented from the rest of the text.

When you select the message or file to include, it is immediately inserted into your message, and you are returned to the create mode.

Including a file works best for straight text files. If you are mailing an application data file, do not include it in your message as its format may not be compatible with **SCO Shell Mail**. Send application data files as separate attachments (page 60).

Including information from other applications in a message

You can insert information in a message from other applications that support the SCO Shell clipboard. The clipboard acts as a temporary storage location for transferring information between applications. To put something on the clipboard see the discussion on using SCO Shell in the *Operating System User's Guide*.

While creating a message, position the cursor where you want the included information to appear. Press ⟨Esc⟩ and select **Paste** from the **Transfer** menu. Select an item from the clipboard.

Note that a non-text item is automatically converted to text. To retain the original file format, send it as an attachment.

Attaching a file to a message

You can attach a file or a clipboard item to an outgoing message. The attachment retains its original file format. This means the recipient can load it back into the application that created it. This enables you to send spreadsheets and graphics through the mail. The attachment is transmitted with the message, but is not part of it. You can only have one attachment per message.

You cannot attach a mail message to another mail message, unless you copy the second message to the clipboard and attach it as a clipboard item. It is easier to include the message (page 59) instead.

To attach a file or clipboard item to your message, press ⟨Esc⟩ and select **Attach** from the **Create** menu.

You can remove the attachment or substitute a different one at any time before you deliver the message, by blanking out or changing the "Attachment Name" and "Attachment Type".

Sending a message to a list of people

If you frequently send mail to a certain group of people (all the people in your department, for example) you can create an "alias", or list of all their login names. You can use the alias when addressing a message to them as a group. Treat an alias just like a user's login: put it in the "To", "Cc", or "Blind Cc" field.

There are two types of aliases: private aliases, which are created by you for your own use, and system aliases, which can be used by everyone.

You can edit, delete, and create new private aliases by selecting **Alias** from the **Options** menu. You can look at system aliases, but you cannot create or change them.

To create a private alias or change the ones that you have, select:

Options ⇨ Alias

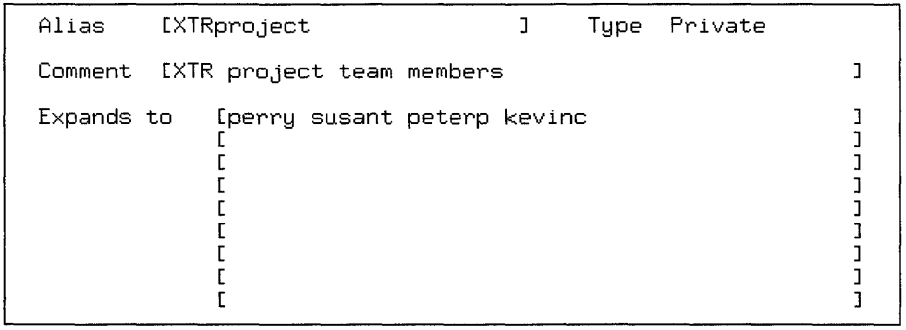
A list of your private aliases is displayed. If the alias list is too big to be shown in its entirety, you can scroll backwards and forwards through it with your cursor movement keys. You can also use the ⟨F5⟩ search key.

Creating a new alias

To create a new alias and add it to your private alias list, select:

Alias ⇨ Add

Enter the details of the new alias on the displayed form:



Field	Description
Alias	Enter the name of the new alias.
Type	Although there are two types of aliases, you can create only private aliases. Therefore, this field is filled in automatically.
Comment	Enter a short description of the alias. This description appears next to the alias name on the alias list.
Expands to	Enter the logins of the people who should be on the alias. The names must be separated by spaces.

Editing an alias

Edit an alias by selecting **Edit** from the **Alias** menu. You can change the name, and add or delete people from the alias.

Select an alias. The same form appears on your screen that you use to create a new alias, but it has the alias information already filled in. You can change any of the fields.

Deleting an alias

Delete an alias by selecting **Delete** from the **Alias** menu. The list of aliases on the screen becomes a point-and-pick list. Move the highlight to the alias that you want to delete, and press ⟨Enter⟩. The alias disappears from the list, and the **Alias** menu reappears. You can delete only one alias at a time.

Viewing an alias

Look at the contents of an alias without changing it by selecting **View** from the **Alias** menu. You can view system and private aliases.

When you select **View** from the **Alias** menu, the list of aliases on the screen becomes a point-and-pick list. Highlight the alias that you want to view, and press ⟨Enter⟩.

When the **Alias** menu is on the screen, you normally see a list of your private aliases. To see a list of system aliases, select **Display**.

If you select **System**, the onscreen list of your private aliases is replaced with a list of all system aliases. When the **Alias** menu reappears, select **View** to examine any of these aliases. Because these are system aliases, you can only look at them; you cannot edit them, delete them, or create new ones.

Including yourself in an alias

By default, when you send mail to a system alias of which you are a member, a copy of your mail is not distributed to you. To receive a copy of such mail, select **Preferences** from the **Options** menu, and then select "Include Self".

Problems with using an alias

There are problems with using system and personal aliases in SCO Shell Mail. If SCO Shell Mail is being run under a German locale, the "Selecting Recipients" feature, which lists system and personal aliases, causes SCO Shell Mail to exit without warning. This problem prevents viewing and selection of all aliases. The "Selecting Recipients" feature is invoked when you enter the ⟨F3⟩ key to fill out the "To:" or "Cc:" header fields of the message you are creating. To work around this problem, run SCO Shell Mail with a non-German **LANG** set, or type the aliases into the header fields manually.

Any attempt to add, modify, view or delete personal aliases causes SCO Shell Mail to hang. This problem occurs in all locales, and there is no workaround.

Verifying the delivery address before sending the message

If you want to check that the delivery addresses of messages exist, select:

Options ⇨ Preferences

Move the highlight to the "Verify Address Before Delivery" field, and press the ⟨Space⟩ bar. An asterisk indicates that the field is selected.

Attaching a signature to your messages

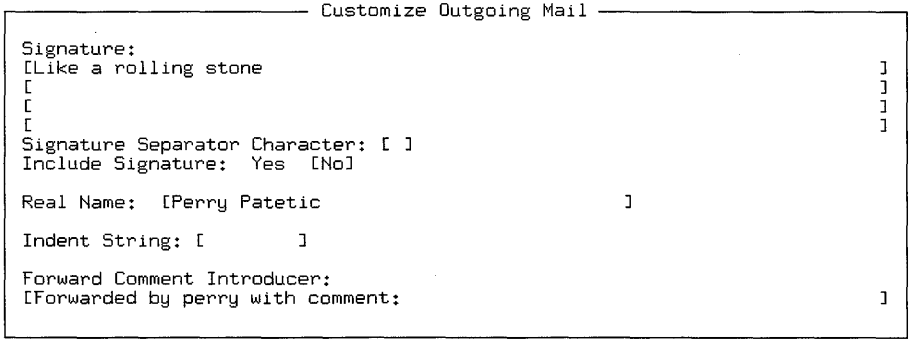
You can create a standard signature notice for the messages that you write. The signature serves the same purpose as the letterhead on a piece of business correspondence, and can include your name, address, business name, and other pertinent information. It differs in that it appears at the end of a mail message.

If you decide to use a signature, **SCO Shell Mail** attaches it to the end of each mail message that you send out. You do not see the signature when you create the message, but the message recipients see it.

To create a signature, select:

Options ⇨ Customize

Your signature can be four lines long. Enter the signature information in the form's four "Signature" fields. Make it look just as you would like it to look at the bottom of your mail messages:



If you want your signature to be separated from the rest of the message by a line, enter any character that you want in the "Signature Separator Character" field (for example, a dash " - ", an underscore " _ ", or an equals sign " = "). **SCO Shell Mail** inserts a full line of those characters above your signature.

After making your signature, select **Yes** from the "Include Signature" field. If you select **No**, **SCO Shell Mail** records your signature but does not add it to messages.

You can return to this form at any time to change your signature, or to turn the signature feature on or off using the "Include Signature" field.

Including your real name in the message header

When you read a message, the header information at the top of the message includes the sender's login; this is usually an abbreviation of the sender's real name. If you would like people to see your full name at the top of messages that they receive from you, enter your real name or title in the "Real Name" field.

Your login still appears at the top of the message, but now it is followed on the same line by your real name, in parentheses.

Running UNIX commands from within SCO Shell Mail

You can run a UNIX command within **SCO Shell Mail**. Type an exclamation mark "!", the command to execute, and press ⟨Enter⟩. When the command terminates, **SCO Shell Mail** will resume.

Sending large files via mail

There is a limitation on the size of message you can send over the mail system. If, for example, you want to include a large file (above 32KB) in a message, the recipient may find that part of the file has been lost (truncated) during the transfer, although no error message will appear. You can find out the size of a file by entering:

!l ***filename***

See the discussion on working with files and directories in the *Operating System User's Guide* or the **ls**(C) manual page for more information about the **l** command.

One solution is to compress and encode a file before including it in your message. Compressing a file reduces its size by about 50%; encoding the file increases the size of the compressed file by about 30%, so you end up with a compressed and encoded file that is about two thirds the size of the original. If this still leaves a file that is too large to send, you should, if possible, split the file into smaller files before compressing and encoding them.

The reason why you should encode a compressed file before you send it is because compressed files may include characters that are not recognizable to **SCO Shell Mail**. Although you may be able to send the compressed file, it may well be truncated or incorrectly transmitted by **SCO Shell Mail**.

If you are a **SCO Shell Mail** user, the encoding of a file is done automatically for you when you attach the file to a message. So, if you want to send a compressed file to someone, you should always attach it to the message instead of including it in the message. If, for some reason, you need to include the file in a message, you should follow the procedure given in "Including information from other applications in a message" (page 59).

Because you can only compress a file from the command line (that is, there is no SCO Shell command for doing this), enter the following UNIX system commands from within SCO Shell using an exclamation mark.

Assume that you have a file called *myfile* that you want to send via mail to one of your colleagues. To compress the file, type:

!compress -c myfile > myfile.Z

The compressed file is named *myfile.Z*; attach this file to the mail message to be sent, but do not edit its contents.

The recipient of your mail message can recover a copy of the original file by saving the attachment to a file (here named *mmsg*), and entering the following command:

!uudecode -s mmsg > myfile

Sending mail to other computers

Many computer systems are connected through networks. Networking may be achieved through network cables, telephone lines, and satellite links. If your SCO system is part of a network, you can send mail to users on other UNIX systems on the network.

Because there are a large number of possible destinations, and because logins are not unique, it is necessary to send mail using an "address" which specifies not only the recipient's name but the organization, department, and even a specific machine to which it is to be sent. (This is equivalent to a postal address.)

For full details of addressing conventions, see "Addressing mail to users on other systems" (page 21).

Chapter 4

Managing mail with MMDF

The installer of an SCO OpenServer system chooses between the Mail Transfer Agents MMDF and **sendmail**. Use this chapter for mail configuration and administration when your system uses MMDF (the Multichannel Memorandum Distribution Facility). For information about using **sendmail**, see Chapter 5, "sendmail administration".

A newly installed SCO system is configured to deliver electronic mail locally (between users on the system itself). Run the **MMDF Configuration Manager** (page 68) if you want to configure your system to exchange mail with other machines, use a domain name server to deliver mail or redirect mail sent to special system accounts, such as *root*, *mmdf*, or *postmaster*.

For daily administration of your mail system, use the MMDF administration managers (page 68).

Tasks described here include:

- "Adding or removing a mail user" (page 78)
- "Managing mail channels" (page 85)
- "Managing mail hosts" (page 79)
- "Managing mail domains" (page 94)
- "Managing mail aliases and lists" (page 81)
- "Modifying MMDF table parameters" (page 110)
- "How MMDF works" (page 103)

How to start a Mail manager

Start the **MMDF Configuration Manager** or any of the Mail administration managers described in this chapter by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, then double-clicking on the **Mail** icon. Start the **Mail Configuration** or administration managers by double-clicking on its icon in the *Mail* window.

Mail administration managers are provided for:

- Hosts (page 79)
- Aliases (page 81)
- Channels (page 85)
- Domains (page 94)
- Tables (page 110)

Problems with SCOadmin Mail Managers

If you installed SCO OpenServer with the German language supplement, these managers are launched in English:

- **MMDF Host Administration manager**
- **MMDF Channel Administration manager**
- **MMDF Domain Administration manager**
- **MMDF Table Administration manager**

To work around this problem, you must explicitly set the **$LANG** environment variable to **german_germany.8859** or **de_DE.ISO8859-1** before launching SCOadmin.

Running the MMDF Configuration Manager

Run the **MMDF Configuration Manager** to perform the initial MMDF configuration after installation. Subsequently, you can run the **MMDF Configuration Manager** at any time to change the initial mail configuration. You must rerun the **MMDF Configuration Manager** when you make a major change to the system, such as when you connect your machine to a new network or remove it from an existing one.

Before you use the MMDF Configuration Manager:

1. Ensure that the MMDF component of the MAIL package is installed with your SCO system. Use **custom**(ADM) to determine this and to install the component if necessary.
2. If you are configuring mail for two or more computers on a network, you must ensure that the network is configured and that the systems involved can communicate with each other.
3. Configure all communications channels that you plan to use to route mail—see "Managing mail channels" (page 85). The **MMDF Configuration Manager** configures MMDF for UUCP, Micnet, and SMTP. MMDF supports other companies' mail products that provide channel programs, but you cannot use the **MMDF Configuration Manager** to configure MMDF to use these. Contact the vendor that supplied the mail software for information on how to configure their product. For information on configuring UUCP, see "Configuring UUCP" in the *System Administration Guide*.
4. If a password for user *mmdf* does not exist, create one now.

> **NOTE** Perform all mail configuration and administrative tasks while logged in as *mmdf*. The exception to this rule (removing a user) is explained in "Adding or removing a mail user" (page 78).

Start the MMDF Configuration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 68), then double-clicking on the **MMDF Configuration** icon, or by entering this at a command line:

```
mkdev mmdf
```

Fill in the configuration information required. "MMDF configuration information" (page 70) describes the information that you must provide.

Refer to "Unsupported configurations" (page 76), which describes situations in which the **MMDF Configuration Manager** may build an incorrect or incomplete MMDF configuration for your site. If your configuration changes for some reason after running the **MMDF Configuration Manager** (for example, you might add a new UUCP host), you should reexamine this list.

Test the new configuration using the utilities described in "Testing MMDF configuration" (page 77).

MMDF configuration information

The information that you need to provide to the **MMDF Configuration Manager** is:

- The local host name. See "Specifying the host name" (this page).
- The names of the network(s) over which you will exchange mail. See "Specifying networks to use with MMDF" (page 71).
- The format of local user addresses as they will appear in outgoing mail. See "Specifying format for mail user addresses" (page 71).
- The name server lookup method to use if you plan to route mail over TCP/IP (SMTP). See "Selecting name server lookup method" (page 72). **You must have TCP/IP installed and configured** to do this.
- The name of a smart host, if any. See "Specifying where to forward mail for unknown hosts or users" (page 73).
- The directory in which you want each user's mail delivered. See "Specifying the location of user mailboxes" (page 73).
- The user(s) to whom mail addressed to *postmaster* will be delivered. See "Specifying the postmaster address" (page 74).
- The user(s) to whom mail addressed to *root* or *mmdf* will be delivered. See "Redirecting mail for root or mmdf" (page 74).
- **If you have UUCP or Micnet installed and configured** and you plan to use MMDF to route mail over UUCP or Micnet, the full domain names, if any, of UUCP or Micnet hosts. See "Specifying full domain names for UUCP and Micnet hosts" (page 75).

Specifying the host name

The **MMDF Configuration Manager** displays the full name of the local host, which you can modify if necessary. The complete name, including the machine name and any other domain names, (page 95) is known as the "fully qualified host name" (also called the "fully qualified domain name"). For example, if you have this mail address:

```
andrei@volga.mynet.com
```

the fully qualified host name is:

```
volga.mynet.com
```

In this case, *volga* is the machine name, *mynet* describes the local site, and *com* specifies that the machine belongs to a commercial organization.

This table shows examples of other fully qualified host names:

slug.ucsc.EDU	*slug* is a machine at the University of California at Santa Cruz in the *EDU* domain
seismo.css.GOV	*seismo* is a machine at the Center for Seismographic Studies in the *GOV* domain
nessie.edinburgh.ac.UK	*nessie* is a machine at Edinburgh University in Scotland in the academic (*ac*) subdomain in the *UK* domain

The **MMDF Configuration Manager** displays the host name followed by the local (if appropriate), subdomain, and top-level domain name as shown in the previous examples. The domain name refers to just the subdomain and top-level domain name (without the host name). For example, *mynet.com*.

See also:

- "Changing a remote host name" (page 81)

Specifying networks to use with MMDF

The version of MMDF included with your SCO system includes support for TCP/IP, UUCP, and Micnet. The **MMDF Configuration Manager** displays the names of the networks that are configured on the local host and prompts you to select one or more of them for exchanging mail. It then configures MMDF to use the selected networks. For information about how MMDF uses the networks, see "About mail channels" (page 92).

Specifying format for mail user addresses

In the **MMDF Configuration Manager**, choose a the mail address format for all users. Choices include the formats "From: ***user@host.domain_name***" and "From: ***user@domain_name***", and possibly a format with one or more additional level of subdomain information, such as *andrei@yangtze.engr.mynet.COM*.

NOTE Support is not provided for 8-bit user names with MMDF.

The second option "hides" the current host name behind the domain name. When you configure MMDF to hide your host name, MMDF identifies all outgoing mail as coming from a single source (your domain name).

For example, the fully qualified domain name for the host that you are configuring is:

```
volga.mynet.com
```

You can "hide" the host name *volga* behind the domain name *mynet.com*. If you do, when *andrei* sends mail outside the organization, the message appears as if it came from *andrei@mynet.com* instead of *andrei@volga.mynet.com*.

NOTE When you join local machines under a single domain name, you create an administrative domain. Within an administrative domain, all user names must be unique so that mail can be delivered to any person anywhere within the domain without a local machine name in the mail address.

To enable MMDF to route mail sent to the domain to its correct destination within your organization, you must create an alias for each user in the domain, mapping each name to the machine where the user receives mail. For example, if *laurie* receives mail on the machine *poet*, add an alias (page 82) called *laurie* to the *alias.user* database. The single member of the alias would be:

```
laurie@poet
```

See also:

- "Adding or removing a mail alias" (page 82) for instructions on adding aliases
- "Managing mail hosts" (page 79)
- "Registering domain names" (page 97)

Selecting name server lookup method

Select the desired level of name service in the **MMDF Configuration Manager**, by specifying which host names you want MMDF to look up on the name server: all mail hosts, local hosts only, or no name service for mail. If you do not have TCP/IP installed and configured and do not plan to route mail over TCP/IP (SMTP), do not use name service for mail.

A name server is a program running on a TCP/IP network that provides a central database of information, such as Internet addresses and the names of hosts on which people receive mail. To use the name server, you must set up the name server before doing the mail configuration. See Chapter 6, "Configuring the Domain Name Service" in the *Networking Guide*.

If your network is large, with hosts added or removed regularly, or if you use the Internet to exchange mail, using a name server for all mail hosts is recommended. If you exchange mail only with other machines in your domain, mail processing may be faster if you specify host name lookup for local hosts only.

Specifying where to forward mail for unknown hosts or users

If your local computer communicates directly with another computer that has more complete information about the entire mail network, you can configure MMDF to route any mail addressed to a host or user it does not recognize to that machine, known as a "smart host". MMDF recognizes two kinds of smart hosts: smart hosts that provide information about the host names of other machines and smart hosts that provide information about the users on the network.

To configure the local computer to use a smart host, click on **Forwarding** in the **MMDF Configuration Manager**:

- To re-route mail addressed to an unrecognized host name (page 91) to a smart host, under "Unknown Host Forwarding", select **Forward mail for unknown hosts**, and either enter the address of the smart host in the text box, or select **Select Host** and choose the smart host from the list. The default action is for the local computer to return mail addressed to unrecognized hosts to the sender.
- To re-route mail addressed to an unrecognized user name (page 91) to a smart host, under **Unknown User Forwarding**, select **Forward mail for unknown users**, and either enter the address of the smart host in the text box, or click on **Select Host** and select the smart host from the list. The default action is for the local computer to return mail addressed to unrecognized users to the sender.

Specifying the location of user mailboxes

To select the directory to which you want MMDF to deliver each user's mail, click on **Mailboxes** in the **MMDF Configuration Manager**. You can choose to maintain all user mailboxes in */usr/spool/mail* (each mailbox is named using the user's ***login_name***), or MMDF can deliver mail to the file *.mailbox* in each user's home directory.

You can specify another mailbox location by manually editing the */usr/mmdf/mmdftailor* file. See the discussion of the **MDLVRDIR** and **MMBXNAME** parameters in the *mmdftailor*(F) manual page, and the instructions in "Editing MMDF configuration files manually" (page 120).

NOTE If you specify delivery to user home directories, and if you use UUCP, */usr/lib/uucp* must have access permissions set to 755 to ensure that mail can be delivered correctly to user *uucp*. See the **chmod**(C) manual page.

Specifying the postmaster address

On the Internet, people send inquiries about user and host names to the "postmaster" address in that domain. RFC (Request for Comments) 821/822, documents that describe Internet protocols, require that every host provide this reserved postmaster mailbox. See "Obtaining RFCs from the Internet" in the *Networking Guide*. For these reasons, you should designate a user on the local system or within the domain as the postmaster and redirect mail sent to *postmaster* to this person.

To redirect *postmaster*'s mail, click on **Redirection** in the **MMDF Configuration Manager** and enter the user name of the designated postmaster in the **Mail sent to "Postmaster" will be sent to** field. If you want more than one user to receive copies, type the account names, separated by spaces, or click on **Select User(s)** and select from the list.

See also:

- "The postmaster address" (page 85)

Redirecting mail for root or mmdf

The SCO system sends mail about any system problems to the *root* user. When problems occur with mail, the system sends mail to the *mmdf* user. This account is reserved for administering your mail system; unless you log in as *mmdf* regularly, you might not find out about problems with the system. For this reason, it is a good idea to redirect *mmdf*'s mail to another person.

Click on **Redirection** in the **MMDF Configuration Manager** to redirect mail from *root*, *mmdf*, or any other non-user special account, to one or more specified recipients. The original account will not receive copies.

In the **You may select user(s)...** field, type the name of the user to whom the mail will be redirected. If you want more than one user to receive copies, type the account names separated by spaces, or click on **Select User(s)** and select from the list.

Under "Non-user accounts", click on **Add Account(s)** and select one or more accounts whose mail you want to redirect. Click on **Remove Account(s)** to delete accounts from the list.

To redirect mail to a user on another machine, you must type in the user's full address in the **select user(s)** field. For example, if you are the system administrator for a group of machines, you can redirect all mail sent to *root* on those machines to your own system mailbox. With this configuration, you do not need to log into each machine to read *root*'s mail. If you are configuring MMDF on *volga.mynet.com* and your user name is *bob@talk.mynet.com*, redirect *root*'s mail to yourself by entering **bob@talk.mynet.com**.

See also:

- "Specifying the postmaster address" (page 74)
- "Redirecting mail to another user" (page 78)

Specifying full domain names for UUCP and Micnet hosts

If you specify in the **MMDF Configuration Manager** that you want to use UUCP or Micnet to exchange mail, after you finish providing configuration information and click on **OK**, the **MMDF Configuration Manager** displays the names of the UUCP or Micnet hosts it finds. If any of them have full domain names, select that host name, click on **Add**, and enter the "fully qualified domain name". See "Specifying the host name" (page 70) for information about fully qualified domain names.

You may exchange mail with UUCP hosts with fully qualified domain names if your host is not on the Internet, but your site has an agreement with another machine on the Internet to transfer your mail. When people on the Internet send messages to you, they use the ***user@machine.domain*** address format (instead of the ***machine!user*** UUCP format). You then use UUCP to connect to that machine and pick up your spooled mail. For example, if the machine *volga.mynet.com* connects to *slug.ucsc.edu*, which is on the Internet, *slug.ucsc.edu* could be a UUCP host with a full domain name. You must configure UUCP before configuring MMDF to use it (see "Configuring UUCP" in the *System Administration Guide*).

See also:

- "Searching MMDF domain tables" (page 108)

Allowing mail replies between an MMDF and non-MMDF system through Micnet

When a host is running MMDF and is connected to a non-MMDF computer through Micnet (see "Specifying networks to use with MMDF" (page 71)), mail users on the MMDF machine are unable to reply to mail originated on the non-MMDF machine.

To allow mail replies between an MMDF and a non-MMDF system through Micnet, a global alias file must be maintained on each machine. Use **netutil**(ADM) to create a global alias file on the non-MMDF machine and to distribute the file to MMDF machines. On the MMDF machines, use **mmdfalias**(ADM) to convert the global alias file to MMDF style:

1. Copy the non-MMDF global alias file to */usr/lib/mail/aliases*.
2. Log in as *mmdf* and change to the */usr/mmdf/table* directory.
3. Run the commands:

   /usr/mmdf/table/tools/mmdfalias
   ./dbmbuild

NOTE Any modifications to the global alias file must be made using **netutil** and distributed again.

Unsupported configurations

In some situations, the **MMDF Configuration Manager** may build an incorrect or incomplete MMDF configuration for your site. Use the MMDF administration managers to update the configuration after running the **MMDF Configuration Manager** if you:

- Use more than one communications channel to exchange mail with a particular host.

  The **MMDF Configuration Manager** sets up MMDF to communicate with the other host using the first configured channel in this order:

  1. TCP/IP
  2. UUCP
  3. Micnet

  If your host accesses another host using two or more channels, but you do not want to use the first applicable channel to transfer mail, the **MMDF Configuration Manager** does not configure MMDF correctly. See "Channel tables" (page 115).

- Have a host that does not exchange mail with every host on each configured channel.

  The **MMDF Configuration Manager** sets up MMDF to exchange mail with every host on the configured channels. For information on removing hosts from a configured channel, see "Managing mail hosts" (page 79). When configuring UUCP and Micnet, you have the opportunity to remove hosts with which you do not exchange mail.

- Exchange mail with two hosts and want to prevent one host from passing mail to the other host through the local host.

The **MMDF Configuration Manager** sets up MMDF to allow one host to send mail through the local host to the other host. To prevent this, you must set up authorization to prevent the transfer. For information on how to do this, see "Specifying MMDF authorizations" (page 122).

- Have more than one channel of the same type.

  The **MMDF Configuration Manager** lists all the hosts that your host accesses using a particular communications protocol in the same channel. To set up authorization to restrict a host's access on a channel, see "Specifying MMDF authorizations" (page 122).

Testing MMDF configuration

Two utilities are provide to test a new or modified MMDF configuration: **checkup**(ADM), described in "Checking for MMDF problems" (this page), and **checkaddr**(ADM) described in "Testing mail addresses" (this page).

Checking for MMDF problems

Use the **checkup**(ADM) command to examine the full MMDF system on the local machine and report any inconsistencies. Normally, **checkup** reports both problems and correct states; **checkup** marks problems with two asterisk characters (**) and encloses advisory information in square brackets. You can tell **checkup** to report only the problems using the **-p** option.

Here is example output from **checkup**:

```
**  Asterisks indicate potentially serious anomalies.

Tailor file             : /usr/mmdf/mmdftailor
**    Unknown tailor    : 'MLCKTYPE advisor'
```

In this case, **checkup** does not recognize the **MLCKTYPE** parameter **advisor** because it is misspelled (the correct parameter name is **advisory**).

If the permissions on any of the MMDF configuration files are incorrect, **checkup** lists this information.

Testing mail addresses

Use the **checkaddr**(ADM) command to test an individual address for validity. For example, to test the validity of the address, *andrei@volga*, enter:

/usr/mmdf/bin/checkaddr andrei@volga

The **checkaddr** checks the address and displays:

```
andrei@volga: OK
```

This command is particularly useful for testing aliases after you have installed a new alias file.

Adding or removing a mail user

When you add or remove a user from the system (for example, a new or retiring employee), there are several mail-related tasks that may need to be performed:

- Adding or removing the user (this page) from system-wide aliases.
- Redirecting mail (this page) sent to the retired account to one or more other users.
- Deciding what to do with a removed user's mailbox: leave it in place, remove it, or move it to another file (page 79).

When you add a new user to the system, a mailbox is automatically set up for the account in the location specified with the **MMDF Configuration Manager**, as described in "Specifying the location of user mailboxes" (page 73).

Adding or removing a user from system-wide aliases

To add a user to one or more existing system-wide aliases, start the **Alias Administration Manager** (page 81), and select **Subscribe** from the **Users** menu. Type the name of one or more users or select them from the list by clicking on the **Select** button. Select an alias by clicking on it (select multiple aliases by dragging the cursor). Click on **Add** to add the user to the selected aliases. If you change your mind about an alias, you can select it from the "Aliases user subscribes to" box and press **Remove**.

Redirecting mail to another user

When a user's account is retired, you may wish to redirect their mail to another person, such as their manager. To do this, start the **Alias Administration Manager** (page 81) and select **Retire** from the **Users** menu.

Enter the name of the retiring user or click on **Select** and choose from the list. Type one or more account names to which you want to redirect the retired user's mail in the "Redirect mail" field.

This removes the user's name from all system-wide aliases, and redirects their mail.

Removing or moving a retired mailbox

The mailbox from a retired account can be left in place, removed, or renamed. To do this, start the **Alias Administration Manager** (page 81) and select **Retire** from the **Users** menu. Enter the name of the retiring user or click on **Select** and choose from the list. Make your selection under "Disposition of mailbox". If you specify **Rename**, type the name of a file in which to save the mailbox.

NOTE Because of the file permissions on user mailboxes, this is the only mail administration task that must be performed as *root*. To remove or move a mailbox, log in as *root* before running the **Alias Administration Manager**.

Managing mail hosts

Use the **MMDF Host Administration Manager** to:

- Examine existing host parameters by selecting one or more parameters from the **View** menu:

  Network (the network over which you communicate with the host)
  Network Address (the host's network address)
  Address Rewriting (page 80) (address format expected by the host)

  Display hosts on a particular network by selecting the network from the **Select** menu.

- Add or remove a host (page 80).
- Modify host parameters.

Start the Host Administration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 68) and double-clicking on the **MMDF Host Admin** icon.

See also:

- the *mmdftailor*(F) manual page

Adding or removing a mail host

To add a new host to the MMDF configuration or remove an existing one, start the **Host Administration Manager** (page 79).

To add a new host, select **Add** from the **Hosts** menu.

Specify the name in the "Host Name" field. This must be the fully qualified host name.

Select from the list the network you will use to communicate with this host.

The **Host Administration Manager** detects automatically the full address information for delivering mail to this host. For example, this will be a UUCP address for a UUCP host or an IP address for a host that communicates with the network using TCP/IP. For more information on host addressing for UUCP and SMTP channels, see "Mail addressing and delivery" (page 97).

NOTE If your host is configured to use name service (page 72), you cannot add new hosts. You must first use the **MMDF Configuration Manager** to select no name service.

Specifying address rewriting

Selecting **Address Rewriting** specifies that this host uses a specific address format and mail directed to it will be reformatted. You can choose to reformat message headers in Internet style, which uses RFC 822-style addresses (for example, <*@A:B@C*>), or you can choose not to reformat headers.

See also:

- "Mail addressing and delivery" (page 97)

Removing a host

To remove a host from the configuration, start the **Host Administration Manager** (page 79), select the name of the host, and select **Remove** from the **Hosts** menu.

See also:

- "Specifying the host name" (page 70)
- the **uucp**(C) manual page

Changing a remote host name

To change the host name of one of the machines with which you exchange mail, start the **Host Administration Manager** (page 79). Select the name of the host, select **Modify** from the **Hosts** menu, then enter the new host name.

See also:

- "Managing mail hosts" (page 79)
- "Specifying the host name" (page 70)

Managing mail aliases and lists

Use the **Alias Administration Manager** to:

- Examine mail aliases and mailing lists and their members. The existing lists are displayed when you start the **Alias Administration Manager**. To look at the existing members, click on an alias or mailing list name (the members are displayed at the right of the window).
- Add, remove, or modify an alias (page 82).
- Add or remove an alias member (page 83).
- Add, remove, or modify a mailing list (page 84).
- Subscribe users to aliases or mailing lists (page 78).
- Retire a mail user (page 78).

Two types of aliases are important to consider: the postmaster alias (page 85) and, if you hide your host name (page 71) behind your domain name, aliases associating users with a home machine (page 85).

Start the Alias Administration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 68) and double-clicking on the **MMDF Alias Admin** icon.

See also:

- the *mmdftailor*(F) manual page
- "Alias and mailing list tables" (page 116) for information about the tables in which MMDF maintains its system-wide alias information

Adding or removing a mail alias

To add a new alias or remove an existing one, start the **Alias Administration Manager** (page 81).

To add a new alias, select **Add** from the **Aliases** menu, and specify the alias name in the "Name" field. The name may not contain ":" or "," characters. Type one or more alias member names in the "Enter Name" field and click on **Add** (or you can click on **Select** and choose each name from the list). Alias member names may be any valid user names or alias names. You must enter at least one alias member when adding a new alias.

NOTE When using an alias to define another alias, be careful not to create an alias loop, where an alias member is another alias of which the original alias is also a member (for example, alias B is a member of alias A, and alias A is a member of alias B).

Maintaining alias members in a file

As an alternative to entering all alias or list members in the **Alias Administration Manager** *Add Aliases* window, you can maintain the list of members in a file, one member per line. When you add a new alias, in the **Enter Name** field, type:

<filename

The < indicates that the list of alias members are maintained in a file and *filename* is the location of the file. You can edit this file to modify the list of alias members.

Removing an alias

To remove an alias, start the **Alias Administration Manager** (page 81), click on the alias to be deleted, then select **Remove** from the **Aliases** menu.

See also:

- "Redirecting and piping mail" (page 84) for information on using aliases to redirect mail into files or through other programs
- the **checkaddr**(ADM) manual page for information on checking the validity of addresses
- the **list**(ADM) manual page for information on the list processor mail channel

Bypassing aliases

Selecting "Allow Bypass" specifies that placing a tilde (~) before an address causes mail to be delivered to the literal address rather than to any aliases for that address. For example, suppose an alias called *george* exists, directing *george*'s mail to a specific machine on the network:

```
george: george@astoria.vegan.edu
```

If you select "Allow Bypass", you can send mail to *george* on the *astoria* machine by addressing it to:

george

You can send mail to *george* on the local machine by addressing it to:

~george

If you do not select "Allow Bypass", entering the tilde character in front of a mail address has no effect and MMDF searches the alias files first for all mail addresses on outgoing or incoming mail (in the previous example, *george*'s mail always goes to *astoria*).

Specifying public aliases

Selecting "public" specifies that the alias is considered public information and remote sites can determine the alias members using the SMTP **EXPN** command.

Adding or removing an alias member

Use the **Alias Administration Manager** to add or remove new members from an existing alias.

To add alias members, start the **Alias Administration Manager** (page 81) and move the highlight to the name of the alias. Select **Add** from the **Members** menu and enter one or more names of new alias members. The member names may be any valid user names or alias names. You can type each name in the "Enter Name" field and click on **Add**, or you can click on **Select** and choose each name from the list.

To remove alias members, in the **Alias Administration Manager** (page 81) Members list, select the name or names to be deleted (press ⟨Ctrl⟩ and click to select multiple names) and then select **Remove** from the **Members** menu.

Searching an alias or mailing list

At any time while viewing or modifying the contents of an alias or mailing list, you can enter a member name or any string in the "Search" field, and the member list scrolls to that point in the alphabetically ordered list. For example, entering the letter **p** in the "Search" field will scroll the member list to those names starting with "p".

Redirecting and piping mail

When adding an alias member, you can use the forward slash (/) or the pipe character (|) to do more complex processing of mail sent to the alias, such as redirecting messages to files. For example, you can enter any of these lines as an alias "member":

```
"network//usr/spool/log/uucp"
"network|cat -v >>/usr/spool/log/mlog"
"network|/usr/bin/lp -dprinter2"
```

In the first example line, MMDF redirects mail addressed to the alias to a file, */usr/spool/log/uucp*. In the second line, MMDF pipes mail addressed to the alias through the **cat**(C) filter before logging it in the *mlog* file. In the third line, MMDF pipes mail addressed to the alias to the **lp**(C) command for printing. These redirection alias examples use the user and group IDs of the user *network*. Although *network* is appropriate in most cases, you can specify any user named in the */etc/passwd* file on your system.

Maintaining mailing lists

Use the **Alias Administration Manager** (page 81) to add, modify, or remove mailing lists. A mailing list is a special-case alias; each has an "owner" who is responsible for handling mailing list administration, including adding and removing members. Error messages about bounced messages and other problems with the list also go to the mailing list owner instead of the originator of the mail. An advantage to mailing lists is that list addresses are not validated at the time each message is sent; this can save a significant amount of processing time with a large list of recipients.

To add a mailing list, add an alias (page 82), giving it the name you want to assign to the mailing list. When you have added the mailing list members, click on the **Mailing List** button near the bottom of the *Add Aliases* window and assign the mailing list owner by typing the user's login name or selecting a name from the list by clicking on **Select Name**.

The owner of the mailing list can easily maintain the list of members if it is kept in a file (page 82). The mailing list owner must have write permission on the file.

To modify a mailing list, add or remove members as described in "Adding or removing an alias member" (page 83). To change the list owner, replace the existing owner's name in the window with the new owner's name.

To remove a mailing list, follow the instructions in "Removing an alias" (page 82).

The postmaster address

Every host must provide a special *postmaster* mailbox (required by RFC 821/822). You can either define an alias (page 82) that maps *postmaster* to an existing user on the system (a person responsible for mail administration) or you can create a *postmaster* account to which mail inquiries and problems can be directed.

The **MMDF Configuration Manager** provides an opportunity to set up an alias (page 74).

NOTE It is best to map *postmaster* to a local user because the address is simpler and, therefore, more likely to be a valid address. Also, with a local user, delivery of the message does not depend on a network connection that might be malfunctioning and therefore responsible for the original mail problem.

If the person responsible for mail administration at your site changes, change the *postmaster* alias accordingly.

Associating users with a home machine

You can configure MMDF to hide your host name (page 71) behind the domain name. When you do this, all user names within the domain must be unique so that mail can go to any person anywhere in the domain without a local machine name in the mail address.

If you decide to hide the host behind a domain name, map each user in the domain to the machine where they receive mail by adding an appropriate alias. For example, if *laurie* receives mail on the machine *poet*, add an alias (page 82) called *laurie*. The single member of the alias would be:

```
laurie@poet
```

Managing mail channels

A mail channel (page 92) permits a machine to communicate with other machines using a single type of network communications protocol. Use the **Channel Administration Manager** to look at the list of existing mail channels and their associated destination hosts on your system.

Start the Channel Administration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 68), double-clicking on the **MMDF_Advanced_Admin** icon, then double-clicking on the **MMDF Channel Admin** icon.

See also:

- "Adding channels" (this page)
- "Removing a channel" (page 88)
- "Modifying channel parameters" (page 89)
- "Managing mail hosts" (page 79)

Adding channels

To add a new channel, start the **Channel Administration Manager** (page 85). Then:

1. Select **Add** from the **Channel** menu.
2. Specify the "Channel Name". The name must begin with a letter and must contain only letters and digits.
3. Select the "Program". The channel program is used to invoke this channel; it receives mail from **deliver**(ADM) and carries it to its destination on the local machine or across the network to a remote machine. The program must be installed in */usr/mmdf/chans*.
4. Enter a "Description" of the channel. This description is used by certain programs as a display line to explain the function of the channel. For example, the **checkque**(ADM) program uses this description in its status reports.
5. Select the name of the "Table" to be associated with this channel. The table lists the hosts that are accessible on this channel.

   If you specify a new channel table name, the name must begin with a letter and must contain only letters and digits. The new table will be defined using the same "Name" and "Description" as are used to describe this channel. For additional information, see "Channel tables" (page 115).
6. Enter a "Configuration string". This string specifies channel-specific configuration parameters. For example, use this string to specify to a **uucp** channel the number of addresses remote hosts can handle in a single message. See "Channel configuration strings" (page 89).
7. You can select additional options for this channel. See "Specifying channel address parameters" (page 87), "Specifying channel delivery options" (page 88), and "Changing error logging levels" (page 128).

See also:

- the *mmdftailor*(F) manual page
- the **uucp**(C) manual page

Specifying channel address parameters

To specify channel address parameters, when adding a channel (page 86) using the **Channel Administration Manager**, select **Add** from the **Channel** menu, then select **Addressing**:

- Select the **Outgoing Address** format (page 97). This parameter specifies how the addresses in outgoing message headers are reformatted on this channel:

RFC 822	uses RFC 822-style addresses (for example <@A:B@C>)
RFC 733	uses RFC 733-style addresses (for example B%C@A)
None	message headers not reformatted

- Specify the **Addressing Options**:

  Select "Domain Hiding" if the machine cannot handle domains delivered via this channel. The leftmost part of domain names will be used.

  Select "Ignore nameserver timeouts" if the host information for this channel is obtained from a nameserver, and you want timeouts ignored. If you do not select "Ignore timeouts", MMDF uses the delay channel to queue messages that temporarily fail address verification because of unavailability of a name server. When timeouts are ignored, MMDF searches other tables for host information.

- Select a table of known hosts if you have selected "Domain Hiding" and want to provide host information in a table. This table must contain your local host's fully qualified host name.

  If you specify a new table name, the name must begin with a letter and must contain only letters and digits. For additional information, see "Channel tables" (page 115).

See also:

- the **deliver**(ADM) manual page
- the *mmdftailor*(F) manual page
- "Mail addressing and delivery" (page 97)
- "Channel tables" (page 115)

Specifying channel delivery options

To specify channel delivery options, when adding a channel (page 86) using the **Channel Administration Manager**, select **Add** from the **Channel** menu, then select **Delivery** and make your selections:

- Select a "Delivery Mode":

Regular	Mail is queued but not sent until **deliver** runs (can be run manually, via **cron**(C), or as a background daemon).
Background only	The channel can be invoked only by a background **deliver** daemon.
Passive	The channel is used as a pickup channel only.
Immediate	The channel can be invoked immediately without having to batch messages.

- Select one or more "Delivery Options":

Allow pick up	The channel can pick up mail from the remote host.
Allow sending	The channel can send mail to the remote host.
Sort by host	After the deliver program sorts by channel, it will also sort by host. This allows as many messages as possible to be sent to a host before the connection is broken.

- Specify the "Time-To-Live" for undeliverable messages. When there is a connection failure to a host, this value specifies the number of minutes between connection retries. The default value is 2 hours.

When you finish entering the delivery options, click on **Close**.

See also:

- the **deliver**(ADM) manual page
- the *mmdftailor*(F) manual page
- "Customizing mail delivery" (page 98)

Removing a channel

To remove a channel from the "Channels" list, start the **Channel Administration Manager** (page 85). Then click on the name of the channel you want to remove, and select **Remove** from the **Channel** menu.

Modifying channel parameters

To view or modify parameters for an existing channel, start the **Channel Administration Manager** (page 85). Then highlight the name of the channel that you want to view or modify, and select **Modify** from the **Channel** menu. Modify the channel parameters as necessary. See "Adding channels" (page 86) for information about the channel parameters.

Channel configuration strings

MMDF uses channel configuration strings to obtain detail information about the channel. Only the **uucp** and **smtp** channels utilize configuration strings.

Specify configuration strings, when adding or modifying a channel (page 86).

The uucp configuration string

The default **uucp** channel configuration string specifies the number of addresses a host can handle in a single mail command. The default value is 1. The syntax of the string is:

```
naddrs=N
```

N is the specified number of addresses.

UUCP parameters include:

naddrs the maximum number of addresses the **uucp** channel puts in a single **uux**(C) command. The default is 1 for backwards compatibility with older UUCP systems, which can process only one address at a time.

If you know that all sites with which you exchange mail via UUCP can handle more than one address, you can set **naddrs** to a higher value. Alternatively, you can create a second UUCP channel, with **naddrs** set to a larger value, for those UUCP sites that can handle more addresses.

maxlen the maximum length of a **uux** command-line. The default value for **maxlen** is 1024. SCO recommends against raising the value of **maxlen**.

rewrite controls how addresses are rewritten when being used as arguments to the **rmail**(ADM) command on a remote machine.

The possible values are:

rewrite=976	uses RFC 976 address format
rewrite=822	uses RFC 822 address format
rewrite=733	uses RFC 733 address format

The default value for **rewrite** is 976.

When using RFC 976 address format, addresses are rewritten as:

Address	Rewritten as
host1!host2!host3!user	*host1!host2!host3!user*
user%host3%host2@host1	*host1!host2!host3!user*
@host1,@host2:user@host3	*host1!host2!host3!user*

Simpler addresses, such as the most commonly-used *user@host*, are left alone except under RFC 976, where they are rewritten as *host!user*.

When using RFC 822 address format, addresses are rewritten as:

Address	Rewritten as
host1!host2!host3!user	*host1!host2!host3!user*
user%host3%host2@host1	*user%host3%host2@host1*
@host1,@host2:user@host3	*@host1,@host2:user@host3*

When using RFC 733 address format, addresses are rewritten as:

Address	Rewritten as
host1!host2!host3!user	*host1!host2!host3!user*
user%host3%host2@host1	*user%host3%host2@host1*
@host1,@host2:user@host3	*user%host3%host2@host1*

The smtp configuration string

The default **smtp** channel configuration string specifies the SMTP name of the host and the character set to be used. The syntax of the string is:

```
hostname=hostname, charset=[7,8]bit
```

hostname is the name of the host for SMTP protocol. This name is used in "Received:" headers and is required by some SMTP implementations. The default is to assign the value of the MMDF hostname (for example, *mynet.com*).

The value of charset specifies whether messages contain 7- or 8-bit data. Using the 8-bit setting violates the SMTP protocol standard specified by the Internet technical bulletin, RFC 821, (available from the InterNIC Registration Services, or InterNIC; see "Registering domain names" (page 97)). However, the 8-bit setting is widely used in Europe. The default value is 7-bit.

Routing mail for unrecognized hosts

Mail addressed to unknown hosts can be forwarded to a smart host (page 73) using a special mail channel, the **badhosts** channel. All mail directed to this channel is sent to the smart host that you specify via the normal channel that you use to communicate with that host (the channel is selected when you run the **MMDF Configuration Manager**).

For example, someone sends mail to *scott@silly.org* but the local host does not recognize the machine *silly.org*. However, your host uses UUCP to communicate with another machine, *columbia.mynet.com*, which is configured as your smart host. MMDF puts the message to *scott@silly.org* in the **badhosts** channel and then uses UUCP to deliver it to *columbia.mynet.com*. The mail system on *columbia.mynet.com* then determines the correct route to send the message to its destination.

Routing mail for unrecognized users

Mail addressed to unknown users can be forwarded to a smart host (page 73) using a special mail channel, the **badusers** channel. This is useful if you have a large number of people at your site. Instead of maintaining information on all the people at the site on each host, you can maintain this information on one central host (the smart host); each individual host maintains information about the local users only. In this case, any mail addressed to users that the local machine does not recognize is directed to the **badusers** channel. The mail routed on this channel is sent to the smart host via the channel (such as UUCP) that you usually use to communicate with that host (the channel is selected when you run the **MMDF Configuration Manager**).

For example, someone on the local host sends mail to *melissa* but the host does not recognize her as a local user. If you communicate with *moose.mynet.com*, a smart host that contains complete user information, MMDF routes the message to this host. This does not mean that *melissa* actually receives her mail on *moose*, just that *moose* has more information about where *melissa* is located.

NOTE If a user has an account on the local host, but a home account elsewhere, you should define a local alias that will direct the user's mail to the home machine (where the user normally receives mail). If not, the mail will stay on the local machine.

Mail sent to this user is sent via whatever channel the local machine uses to communicate with the home machine, not via the **badusers** channel.

About mail channels

A channel permits a machine to exchange data using a single type of network communications protocol. It handles the mail transport protocol so that neither the operating system nor the rest of the MMDF system has to know about the intricacies of a particular mail transport protocol. Channels act not only as protocol handlers, but in some cases actually initiate the communications to the network or to another machine as needed. They also may convert address or message formats as necessary.

For information about the channel programs themselves, see "Channel programs" (page 93).

Common channels include:

badhosts Called when a specified machine in a mail address is unknown to the local machine to forward the message to a smart host. See "Routing mail for unrecognized hosts" (page 91). The use of "bad" is really a misnomer; any mail addressed to an unknown machine is sent on this channel. **badhosts** uses a different channel, such as **uucp** or **smtp**, to perform the actual mail forwarding.

badusers Called when mail is addressed to an unknown user name to forward messages to a smart host. See "Routing mail for unrecognized users" (page 91). Like **badhosts**, **badusers** uses a different channel, such as **uucp** or **smtp**, to perform the actual mail forwarding.

list Called to remail messages. This channel simply invokes **submit** and feeds the addresses and text back into the MMDF mail system. This is often used to avoid long address validation or to force the validation to occur in the background for very large mailing lists. This also ensures that MMDF sends any problem reports to the list maintainer.

local Called to deliver mail to mailboxes and processes on the local machine.

micnet Called to deliver or accept mail from a Micnet network connection.

smtp Called to deliver or accept mail from a TCP/IP network connection. The **SMTP** channel transfers messages by establishing a TCP/IP connection to a remote machine, and using the Simple Mail Transfer Protocol (SMTP) to send one or more mail messages. The Internet Protocol (IP) allows many local- or wide-area networks to be interconnected transparently. This permits the MMDF SMTP channel to exchange messages with any machine on any network to which it is connected. For example, if your machine connects to

the Internet, you can exchange messages directly with any machine in the world that is also connected to the Internet.

uucp Called to direct mail to UUCP delivery to another machine, or to accept mail from a UUCP connection from another machine. Incoming mail is converted into the format specified by the Internet technical bulletin, RFC 822, available from the InterNIC Registration Services, or InterNIC. See "Registering domain names" (page 97).

Outgoing mail on the UUCP channel includes a "From " line and the mail path arguments are separated by UUCP exclamation point characters (!).

For more information about UUCP, see "How UUCP works" in the *System Administration Guide* and "Configuring UUCP" in the *System Administration Guide.*

delay When you use MMDF in conjunction with the domain name server system, this channel is called to queue messages that temporarily fail address verification because of unavailability of a name server. The **MMDF Configuration Manager** configures this channel automatically.

See also:

- "Channel tables" (page 115)

Channel programs

For each channel, MMDF provides two programs: an input program associated with incoming mail, and an output program associated with outgoing mail. For example, the UUCP channel has */usr/bin/rmail* as its input program and */usr/mmdf/chans/uucp* as its output program.

On the MMDF system, the server is a channel program that monitors the network for incoming mail. On systems other than MMDF, the function of a mail server is built into the mail delivery system. The relationship between a channel and server is shown here in the following graphic.

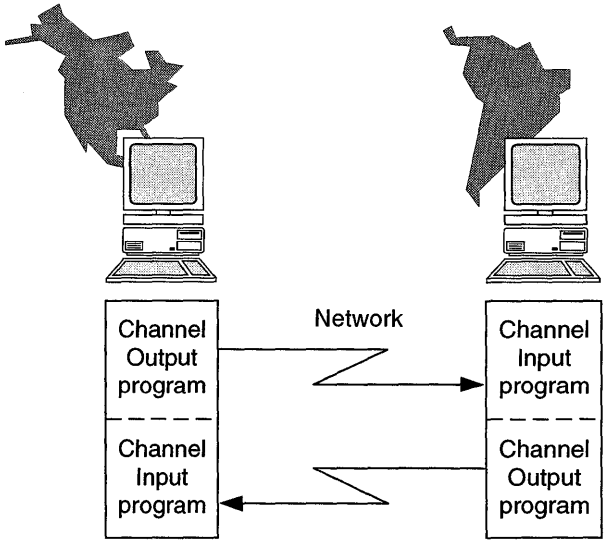


Figure 4-1 Channel and server relationship

Even if MMDF is only being run on one side, the other side, while not correctly called a channel program, performs the same function.

Managing mail domains

Domain information uniquely describes the site where a computer is located and generally includes the machine (host) name, a department (optionally), and the site's organization or country. MMDF uses domain descriptions to deliver mail messages to the appropriate locations. See "Domain names" (page 95). Note that a fully qualified domain name uniquely identifies a machine, but not the path by which messages reach that machine.

Start the Domain Administration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 68), double-clicking on the **MMDF_Advanced_Admin** icon, then double-clicking on the **MMDF Domain Admin** icon. The **Domain Administration Manager** displays a list of existing domains.

Use the Domain Administration Manager to add or remove a domain (page 95) and to view or modify domain parameters (page 95).

See also:

- "Domain tables" (page 111)

Adding or removing a domain

To add a new domain, start the **Domain Administration Manager** (page 94). Then:

1. Select **Add** from the **Domains** menu.
2. Enter an abbreviated name for the domain in the "Domain Label" field.
3. Specify a "Comment", which can describe the purpose of the domain.
4. Enter full name for the domain (this page) in the "Domain Name" field.
5. Specify the Table (page 111), which is the name of the table where the known sites in this domain are listed. Click to the right of the text box for a list of tables.

To remove a domain, click on the name of the domain that you want to remove and select **Remove** from the **Domains** menu.

See also:

- "Domain tables" (page 111)
- the description of the MDMN keyword in the *mmdftailor*(F) manual page

Modifying domain parameters

To view or modify parameters for an existing domain, start the **Domain Administration Manager** (page 94). Then highlight the name of the domain that you want to view or modify, and select **Modify** from the **Domains** menu. Modify the domain parameters as necessary. See "Adding or removing a domain" (this page) for information about the domain parameters.

Domain names

A "domain name" is the section of a mail address that appears to the right of the at (@) character, for example, *mynet.com*. The domain name describes the site where a machine is located and generally includes the machine (host) name, a department (optionally), and the site's organization or country. MMDF uses the domain name to deliver the message to the appropriate location. Domain names can be either upper or lowercase; MMDF is case-insensitive when evaluating domain names. Note that the domain name uniquely identifies a machine, but not the path by which messages reach that machine.

This is the convention for specifying domains:

hostname.subdomain.top-level

If the domain includes a department, the convention is:

hostname.local.subdomain.top-level

Here is a description of each of the domain levels:

Top-Level Domain

A top-level domain is an officially registered name that describes the purpose of a group of institutions or a code that is associated with a country.

You can only use registered top-level and subdomain names if you have registered your organization with SRI International. For information on registering your domain, see "Registering domain names" (page 97). If you have not registered with SRI, use the UUCP top-level domain.

In the United States, the common top-level domains on the Internet are:

COM	commercial institutions
code	country code

NOTE The International Standards Organization (ISO) standard 3166 defines the country codes. For example, *US* is the country code for the United States, *AU* for Australia, *DK* for Denmark, and *JP* for Japan.

EDU	educational and research institutions
GOV	government institutions
MIL	military institutions
NET	network
ORG	organization (generic)
UUCP	a domain name where users transmit information between cooperating neighbor machines via UUCP

Subdomain

An officially registered name that describes a company, department, or any subgroup under a top-level domain; *sco* is an example of a subdomain in the domain *COM*.

Local Domain

A name recognized only within an organization that has meaning only within that organization; a department name such as *engr* is an example of a local domain.

Registering domain names

If you or any user at your site plan to receive or send mail outside of your organization, you should register a top-level domain or subdomain with the InterNIC Registration Services. Even if you are only using UUCP, registration with the InterNIC is your only guarantee that the name of your site is unique.

To register a top-level domain and subdomain name call or write to the Network Information Center (NIC) standards organization at this address:

Surface mail:	Network Solutions Attn: InterNIC Registration Services 505 Huntmar Park Drive Herndon, VA 22070
Telephone:	1-703-742-4777 (7:00 AM to 7:00 PM EST)
Email address:	*hostmaster@internic.net* - host, domain, network changes, and updates *action@internic.net* - computer operations *mailserv@rs.internic.net* - automatic mail servers
Network address:	198.41.0.5 (RS.INTERNIC.NET)

The earlier your site enrolls a domain name and InterNIC Registration Services gives you an address, the less likely it is that you will have to alter a machine name or other site identifier later.

Mail addressing and delivery

MMDF expects addresses to conform to the RFC 822 mail standard, available from the Network Information Center (this page) at SRI International. MMDF also supports RFC 733-style addresses for compatibility with earlier mail systems.

In addition to those in RFC 822, the following address delimiters are recognized within the local part of addresses (in order of precedence):

@ % !

The "!" delimiter is interpreted as "host!user" while the others are interpreted as "user?host". The following table shows examples.

Address	Queued as
a.b!user%c@localhost	*a.b!user@c*
a.b!user@localhost	*user@a.b*
user%a@localhost	*user@a*

Referencing addresses maintained in a file

Addresses can be maintained in a file and indirectly referenced, using one of these file specifications:

<filename
:include:filename

The angle bracket must be the first non-blank character of the specification.

Addresses in the file may be separated by commas or newlines.

Customizing mail delivery

The **deliver**(ADM) program manages mail delivery under MMDF. To customize deliver processes (page 100), edit the MMDF system startup file, */etc/rc2.d/S86mmdf*, while logged in as *root*. You can:

- change the frequency with which **deliver** runs.
- change the number of **deliver** processes that run.
- create separate **deliver** daemons for each configured channel.
- specify a different time interval for each **deliver** process.

Changing the deliver interval

To change the interval between **deliver** processes from the default of every 10 minutes, add the **-T** option to the command line in */etc/rc2.d/S86mmdf*. For example, set up your system so that **deliver** runs every 60 seconds using this **deliver** startup line:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -T60"
```

NOTE When you start **deliver** with the **-b** option only, *one* **deliver** process manages all configured channels. The **deliver** daemon periodically checks each channel's queue for mail to deliver. This is known as "sweeping the queues".

Running multiple deliver processes

When you have a single **deliver**(ADM) program managing a number of channels, **deliver** goes through the channels individually and tries to deliver all the messages in a channel's queue before going on to the next channel. You can instead configure your system to start multiple **deliver** programs, each servicing a single channel. In this case, the **deliver** daemons work in parallel. This is a desirable configuration for a mail gateway machine because it increases the overall mail bandwidth of the machine.

NOTE For each **deliver** process that you initiate, system overhead increases.

To start separate **deliver** processes to manage each channel, add **deliver** startup lines for each channel to the */etc/rc2.d/S86mmdf* file. Use the **deliver -c** option to specify each channel. For example, to create separate **deliver** processes for local, UUCP, Micnet, and SMTP channels, put these lines in *S86mmdf*:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -clocal"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -cuucp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -cmicnet"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
```

If you have a significant amount of mail traffic on a single channel, you can start more than one **deliver** process on that channel by adding more than one **deliver** startup line for the channel. For example, to start three **deliver** processes to sweep the queue for the SMTP channel, add these lines:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
```

Keep in mind that each **deliver** process that you start increases the load on the system.

Specifying different time intervals for deliver

The **MSLEEP** parameter in the *mmdftailor*(F) file determines the default time interval for the **deliver** daemon to sweep through the queues for all the channels. The default setting for **MSLEEP** is 600, meaning that **deliver** sweeps the queues every 600 seconds, or 10 minutes. We recommend that you set up **deliver** to run every 60 seconds. To use this interval for all the channels, change the value of **MSLEEP** in *mmdftailor* to 60.

If you create a separate **deliver** daemon for each channel, you can set up a different time interval for each **deliver** process by by adding the **-T** option to each **deliver** startup line in */etc/rc2.d/S86mmdf*. For example, to run **deliver** on the local channel every 30 seconds and on the UUCP channel every 5 minutes, put these lines in *S86mmdf*:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -clocal -T30"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -cuucp -T300"
```

The deliver program

The **deliver**(ADM) program manages message delivery under MMDF by moving mail from the queue to the appropriate channel. To do this, **deliver**:

1. determines the list of channels to process. You can specify the exact list on the **deliver** command line. The default is to process all nonpassive channels.
2. determines which messages to process. You can specify the exact list on the **deliver** command line. The default is to process all messages in the queue for each channel.
3. sorts the messages in order of channel, submission time, and (optionally) host. The message list is not sorted (processed in order read) if it is specified on the command line or if the number of messages exceeds **MAXSORT**, as specified in the *mmdftailor* file.

The submit program

The **submit**(ADM) program determines the destination of a message, designates the channel to use to deliver the mail, and places the message in the appropriate channel queue. To do this, **submit**:

1. accepts incoming mail from a channel.
2. uses the message's "To:" header, command-line syntax, and the MMDF configuration files (page 109) to build the fully qualified domain name (page 95) for the destination host.
3. uses the fully qualified domain name to determine the appropriate channel through which to send the message, and places the message in the appropriate channel queue.

Specifying uux routing options for UUCP

The **uux**(C) program can be used to send files between machines over UUCP or to execute commands remotely. If you use UUCP to route mail, you can specify the options to be used when sending files and executing commands on the remote machine. By default, MMDF uses these options to **uux**:

```
uux - -r
```

This setting specifies that UUCP queue the job until the next time **uucico**(ADM) (the UUCP file transport program) runs rather than transfer it immediately.

To change this, use a text editor to add a **UUXSTR** parameter to (*/usr/mmdf/mmdftailor*). For example, if you want **uucico** to run immediately, add this line:

```
UUXSTR "uux -"
```

In general, we recommend that you use one of these **UUXSTR** settings:

UUXSTR setting	Describes
UUXSTR "uux -"	Run **uucico** immediately
UUXSTR "uux - -r"	Queue job until **uucico** next runs
UUXSTR "uux - -gA"	Run **uucico** now and transmit job early in session
UUXSTR "uux - -r -gA"	Queue job, then transmit early in next **uucico** session

See also:

- the *mmdftailor*(F) manual page
- "Editing MMDF configuration files manually" (page 120)

Specifying the MMDF "signature"

To change the signature that MMDF uses when notifying senders of mail delivery problems, you must login as *mmdf* and edit */usr/mmdf/mmdftailor*. Add a line (anywhere in the file) defining text for the **MSIG** parameter. The message you define should indicate which mail routing system was responsible.

For example:

```
MSIG     "MMDF Mail System"
```

With this definition, when users receive failed mail messages from MMDF, the message appears to be from "MMDF Mail System".

Forwarding mail from one account to another

To forward mail sent to one account to a second account, create a file called *.maildelivery* in the first user's home directory. The file must be owned by the user or by *root*. Use this command to set the file's permissions:

```
chmod 600 .maildelivery
```

Suppose you want to forward mail addressed to *peter* to user *wendy*. Create *.maildelivery* in *peter*'s home directory. Place this line in the file:

```
addr  peter  |  A  /usr/lib/mail/execmail  wendy
```

Mail sent to *peter* will now be sent to *wendy*.

You can create a system-wide forwarding file (with the same format as *.maildelivery* files) in */usr/lib/mail/maildelivery*. Any entries in a user's personal *.maildelivery* file take precedence over those in the system-wide file.

This same task can be accomplished using the **Alias Administration Manager**. See "Managing mail aliases and lists" (page 81).

NOTE Mail forwarding can also be defined by placing the forwarding address in a *.forward* file in the user's home directory. The *.forward* file must be readable by user *mmdf*. However, this definition is overridden if there is any system-wide alias defined for the user.

See also:

- the *maildelivery*(F) manual page

Changing the system name

To change the system name:

1. Log in as *root*.
2. Modify the system name using the command:

   `uname -S` ***newname***

   where ***newname*** is the new system name.
3. Relink the kernel by entering:

   `./link_unix`

   This will keep the nodename and the system name consistent.
4. Run **mkdev mmdf** and change the host name at the top of the window.
5. If you have SCO TCP/IP installed and configured, make these changes:

   Edit */etc/hosts* and change the line corresponding to the old system name to read:

   IP_address ***newname*** ***newname.company.COM***

   where ***IP_address*** represents the IP address of this system. ***newname*** is the new system name and ***newname.company.COM*** is the fully qualified domain name.
6. Edit */etc/default/nbconf* and change the value of **NB_HOST** to ***NB_HOST=newname.***
7. Reboot the system.

How MMDF works

MMDF provides transparent access to the different networks and mail transport protocols, regardless of the Mail User Agent (MUA) employed. One or more MUAs, such as **mail**(C) or **scomail**(XC), can be used on a single system.

For information on how MMDF works, see:

- "Processing incoming mail" (this page)
- "Processing outgoing mail" (page 105)
- "How MMDF routes mail" (page 107)
- "MMDF configuration files" (page 109)
- *mmdftailor*(F)

Processing incoming mail

MMDF receives incoming mail from each channel's input program. How MMDF handles incoming mail is illustrated in Figure 4-2, "Incoming mail" (page 104).

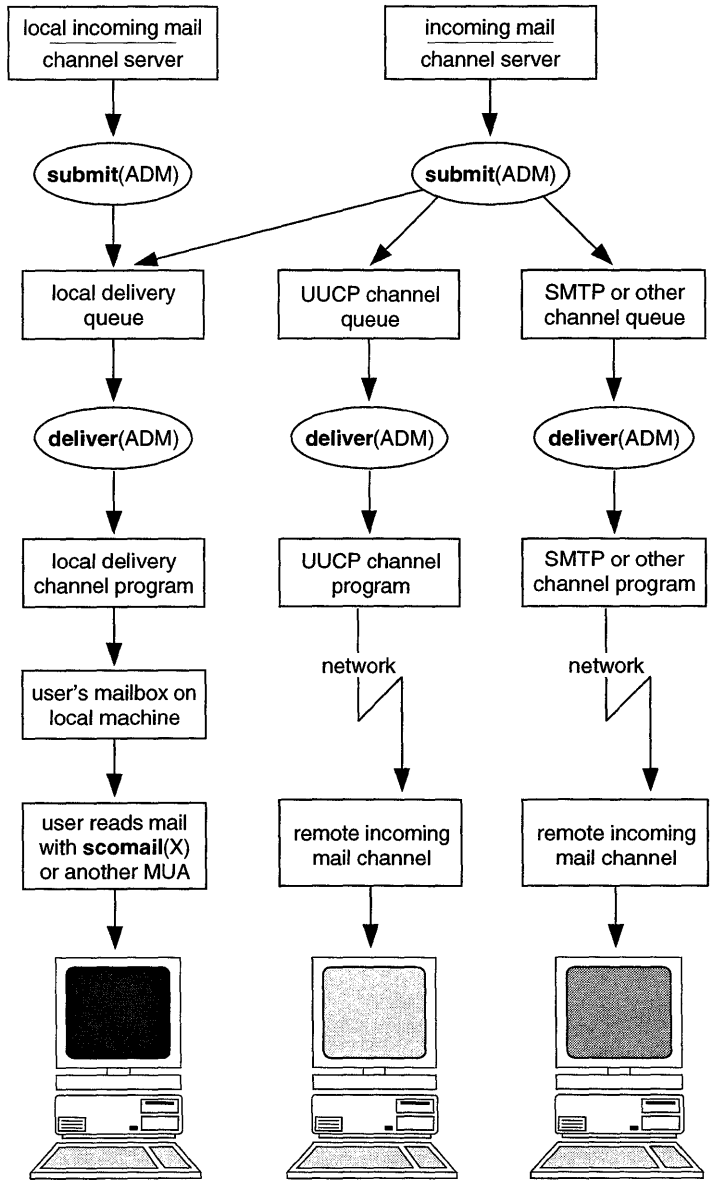


Figure 4-2 Incoming mail

Each message is delivered using this procedure:

1. The **submit** program (page 100) determines the destination of the message and then designates the channel to use to deliver the mail.
2. MMDF selects a channel:
   - Mail destined for the local machine goes to the **local** channel.
   - When the machine is local, but the user is not, mail goes to **badusers**.
   - If mail requires further processing to determine the destination, the message goes to the **badhosts** channel.
   - If the message is destined for another machine, such as when the local machine is acting as a gateway between networks, MMDF selects another network channel, such as UUCP.
3. The **submit** program puts the message in the appropriate channel queue.
4. The **deliver** program transfers the mail from each queue to the appropriate channel.
5. The message is delivered to the user.

Processing outgoing mail

Outgoing mail starts when the user invokes an MUA, such as **mail**(C) or **scomail**(XC) to compose a mail message, or when incoming mail is accepted for routing through the local host to another destination. How MMDF handles outgoing mail is illustrated in Figure 4-3, "Outgoing mail" (page 106).

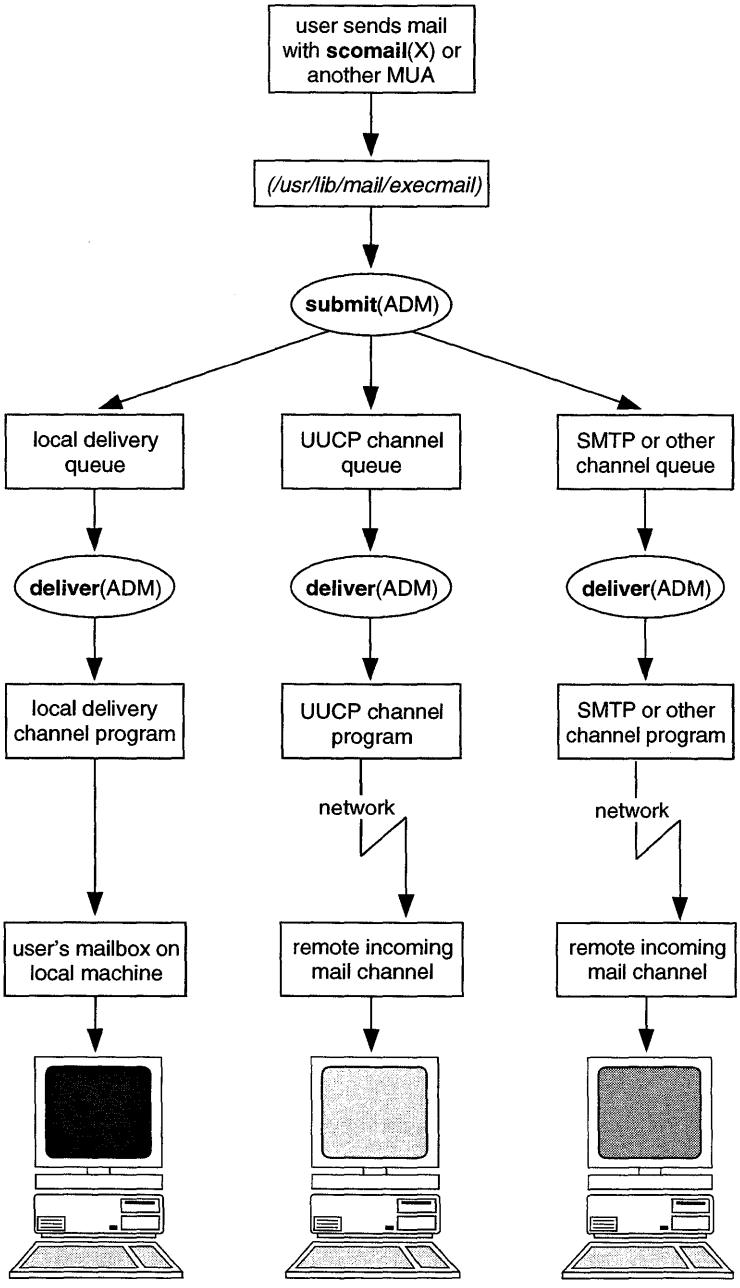


Figure 4-3 Outgoing mail

A mail message is sent using this procedure:

1. When the user sends the messages for delivery, the MUA ensures that the message has the required mail headers (page 121).
2. The MUA transfers the message to **execmail**, which sends it to **submit**(ADM). Alternatively, **submit** may accept the incoming mail from a channel.
3. The **submit** program builds the fully qualified domain name (page 95) for the recipient by mapping the information in the "To:" field with the information in the domain tables.
4. **submit** then looks up the fully qualified domain name in the channel tables (page 115) to determine the appropriate channel through which to send the message, and places the message in the appropriate channel queue.
5. The **deliver**(ADM) program, which runs periodically (by default, **deliver** runs every 10 minutes), transfers the message from the channel queue to the appropriate channel program, which routes it to its destination.

For example, you send a message with this "To:" line:

```
To: andrei@mynet.com
```

The **submit** program looks in the appropriate MMDF configuration table for *mynet* and builds the fully qualified domain name; for example:

```
volga.mynet.com
```

Now, **submit** determines the appropriate channel to use to send the message. In this example, all messages in the *mynet.com* domain are sent to the outside world on the UUCP channel, so **submit** puts the message in the UUCP channel queue. Then **deliver** picks up the message and passes it to the channel program. The channel program transfers the message out of the MMDF mail system and into the UUCP subsystem where it is queued and sent via UUCP.

How MMDF routes mail

MMDF uses the information in the configuration files to route mail on your system. Note that MMDF never searches the alias, channel, and domain tables directly, but instead searches the hashed *dbm* database. See "Rebuilding the hashed database" (page 121).

Mail arrives at and leaves computers using one of several different methods (such as UUCP, or TCP/IP) called "channels". The MMDF **submit**(ADM) command accepts the incoming mail from a channel and determines the correct outgoing channel to use based on the destination host. The **submit** program uses the information in the domain tables to map the way the incoming mail describes the destination host to the way the host recognizes that destination host. Based on the host description, **submit** uses the channel tables to deter-

mine the outgoing channel to use to route the message and places the message in the appropriate queue.

Then, using the information from */usr/mmdf/mmdftailor*, **deliver**(ADM) moves the mail from the queue to the appropriate channel. For example, if **submit** places a message in the UUCP channel queue, **deliver** moves that message to the UUCP channel. The **deliver** command can also place mail in a channel and let another program (such as **uux**(C) for the UUCP channel) carry out additional steps to resolve the circumstances dictated by the type of channel. The channel program sends the mail across the network to the proper destination.

Searching MMDF domain tables

The **submit**(ADM) command uses the domain tables for two purposes: to specify the fully qualified host name and (optionally) to specify the route to a host by listing the fully qualified host names of one or more intermediate hosts through which mail is to be routed.

First, **submit** separates the fully qualified host name into two parts: the name of the domain table and the host name to match on the left-hand side of the entries in the domain table. For example, in the address *david@engr.canada.com*, the name of the domain table to search is *canada.com* and the host name to search for is *engr*.

MMDF tests an address for matches against the domain names in the **MDMN** entries in *mmdftailor*. For example, the address *david@engr.canada.com* matches this **MDMN** entry:

```
MDMN   "canada.com", show="Canada Delivery", tbl=canadadom
```

Then, MMDF searches *mmdftailor* for the **MTBL** entry for *canadadom*:

```
MTBL  canadadom,  file="canada.dom",   show "Canada Delivery"
```

MMDF uses the file associated with *canadadom* (*canada.dom*).

Thus, to route our message to *david@engr.canada.com*, **submit** uses this algorithm to search the domain tables for a match in the address:

1. Search for the host name (*engr*) in the left hand side of the *canada.com* domain table.
2. Search in the left hand side of the domain table (in this case, *com*), if it exists, for the domain name (*engr.canada*).
3. Search the left hand side of the *root* domain table for the fully qualified host name (*eng.canada.com*).

4. Search the relevant tables that include ***flags=route*** in the **MTBL** line in *mmdftailor* for substrings of the address. For example, search the *com* table for *canada*; search the *root* table for *canada.com* or *com*.

   The ***flags=route*** parameter in the **MTBL** entry enables that table entry to match addresses for an entire subdomain, acting as a gateway. Do not use ***flags=route*** on tables other than the *root* domain table unless you have internal subdomains.

5. Search the tables that include ***flags=partial*** in the **MTBL** line in *mmdftailor* for the input name, in this case, the fully qualified host name (*eng.canada.com*), regardless of the domain name.

   Use the ***flags=partial*** parameter on the *local* domain table so that users do not have to specify the full domain to send mail on the local machine.

If **submit** finds no match at all, as a last resort, it uses the **badhosts** channel, if it exists. Because MMDF uses the first domain that has an exact match without looking for other matches in later tables, the order in which you list **MDMN** definitions is significant. Make sure the local domain table appears first and the *root* domain table is last in *mmdftailor*.

If the address provided is ***username@host*** (for example, *david@engr*), MMDF performs the same search as described above except that there is no *canada.com* to match in the **MDMN** entries in *mmdftailor*. In this case, MMDF searches the *root* domain table and then the tables that include ***flags=partial*** in the **MDMN** line.

MMDF configuration files

MMDF uses a variety of configuration files, including:

- Authorization log files (page 129)

  Logs of authorization checks and results. You can specify how much information you want logged.

- Channel tables (page 115)

  MMDF uses channel tables to determine which channel to use and to determine address information.

- Domain tables (page 111)

  MMDF uses domain tables to map hosts to domain names and to obtain information about routing.

- Alias tables (page 116)

  MMDF uses alias tables to obtain information about aliases and their members.

- *mmdftailor* file (page 118)

  The *mmdftailor* file defines all the MMDF mail attributes for the local computer.

These files are all ASCII text and may be modified with a text editor. When adding long lines, you can use a backslash (\) as a line-continuation character.

When you perform the initial MMDF configuration, the **MMDF Configuration Manager** sets up the appropriate table and *mmdftailor* files.

Modifying MMDF table parameters

To view and modify the table parameters listed in this section, start the **Tables Administration Manager** by opening the *Mail* window as described in "How to start a Mail manager" (page 68), double-clicking on the **MMDF_Advanced_Admin** icon, then double-clicking on the **MMDF Table Admin** icon. Select **Modify** from the **Tables** menu, then enter:

Name — A short name that can be used to describe the table elsewhere, for example when you specify a domain table (page 95).

File — The file from which the contents of the table are built.

Description — A comment, which can describe the purpose of the table.

Table Data — Specifies whether the table data comes from a file (specified under "File" in the window), or by querying a name server (page 72).

Lookup Options

Search other tables specifies that MMDF should search other domain tables for this domain. Use this option for the local domain table so local users do not have to specify the full domain to send mail on the local machine.

Do subdomain lookups specifies that MMDF should search for successive subdomains of this domain if no exact match exists. For example, for the domain *mynet.mynet.com*, search the *com* table for *mynet*; search the *root* table for *mynet.com* or *com*.

Nameservice lookup type

By domain specifies that MMDF should query the name server for the given address using the domain name.

By full name specifies that MMDF should query the name server for the fully qualified domain name to determine the delivery address.

Nameservice Options

Abort on timeout specifies that MMDF should not search other domain tables if it encounters a timeout problem when searching a name server domain table.

See also:

- the description of the MTBL keyword in the *mmdftailor*(F) manual page

Domain tables

MMDF uses domain tables to match a short host name to its fully qualified host name. This allows users to address mail to other machines within a host's domain using just the host's name, instead of requiring each host's fully qualified domain name in addresses. For example, if the domain *mynet.COM* includes machines named *volga, yangtze, mekong,* and *seine,* a user on *volga* can address mail to user *andrei* on *yangtze* as *andrei@yangtze* instead of having to use the fully qualified name, *andrei@yangtze.mynet.COM.*

Domain tables can also convey information to MMDF about how machines are connected, and they can specify special domain routing considerations.

Domain tables are named using ***domain_name.dom*** (except for *root.dom,* which contains any domain information not named in other domain tables). The standard practice is to maintain a separate domain table for the domain in which the host machine resides. Domain tables are created by default in the */usr/mmdf/table* directory (or the directory specified by the **MTBLDIR** parameter in the *mmdftailor* file).

The SCO system includes two domain tables in the */usr/mmdf/table* directory:

Domain table	Domain	Describes
local.dom	local	machines in the local domain
root.dom	root	domains not listed in other domain tables

NOTE The *root.dom* table also contains information about how to access top-level domains, such as *MIL* and *GOV*. (The name *root* implies the top level of a hierarchy.)

Use the **Domain Administration Manager** (page 95) to create new domain tables for each domain. For example, create a domain table for the *mynet.com* domain and name it *mynet.dom*. You do not need to use the **.dom* naming scheme; however, this filename extension makes it easier to determine the purpose of the file.

Domain table format

Each domain table consists of at least two columns of information: the first (leftmost) column lists the host name and the second column lists the fully qualified name for that host. The domain names can be either upper- or lowercase. Use tabs, spaces, a colon, or a combination of these characters to separate the columns. Additional columns indicate names of one or more forwarding machines (machines through which mail should be routed for delivery to the host listed in the second column).

The name of a domain table determines the domain names for which MMDF searches. For instance, the domain table for UUCP generally contains entries for names in this form:

hostname: ***hostname***.UUCP

However, you can create an entry to map a specific UUCP address to another address. For example, to map *research.uucp* to *research.jcn.com*, the entry in the *uucp.dom* table looks like this:

```
research:       research.jcn.com
```

The following sections give examples of different domain tables.

The local.dom table

The *local.dom* table describes the machines in the local domain (independent of the channel that MMDF uses to reach each machine). For example, if there are four machines in the domain *mynet.com*, the *domain.dom* table looks like this:

```
volga:          volga.mynet.COM
yangtze:        yangtze.mynet.COM
mekong:         mekong.mynet.COM
seine:          seine.mynet.COM
```

The *domain.dom* table maps each machine to the fully qualified host name in the *mynet.com* domain.

The root.dom table

The *root.dom* table defines any the hosts and domains not defined in the other domain tables. This table can also contain information about how to access top-level domains, such as *MIL* and *GOV*. For example, if your host connects to the UUnet network system, you can set up MMDF to send all mail addressed to specific domains to *uunet.uu.net*. To do so, set up your *root.dom* table like this:

```
COM:    uunet.UU.NET
EDU:    ucscc.EDU
MIL:    uunet.UU.NET
GOV:    star.GOV
NET:    uunet.UU.NET
```

In this example, all mail directed to the *com*, *mil*, and *net* domains is sent out on UUnet; mail directed to the *gov* domain goes to *star.gov*; and mail sent to the *edu* domain goes to *ucscc.edu*. This includes mail addressed to domains that are contained within the named domains. For example, mail sent to *user@cs.berkeley.edu* is sent to *ucscc.edu* unless a separate domain entry exists for *cs.berkeley.edu* or for *berkeley.edu*.

The uucp.dom table

The *uucp.dom* table is created if UUCP is configured on your system and if you add any hosts with a fully qualified name of ***hostname.UUCP*** . This table describes the hosts in the UUCP domain. For example, if your host connects to the remote machines *cactus* or *palm* via UUCP, the *uucp.dom* table looks like this:

```
cactus: cactus.UUCP
palm:   palm.UUCP
```

In this case, MMDF directs any mail sent to the *cactus* or *palm* machines to the UUCP network.

Creating other domain tables

Use the **Domain Administration Manager** (page 95) to create new domain tables. When you subsequently add new hosts that belong to that domain, they are added to the appropriate table. For example, if you create the domain table *rivers.dom* and then add a host named *nile.rivers.COM*, this entry is added to *rivers.dom*:

```
nile:   nile.rivers.COM
```

LAN considerations

If you are configuring MMDF for use on a local area network (LAN), you can use the domain tables to distribute the processing load on the machines, or you can designate a special mail server machine to route all the messages.

You have these choices:

- MMDF Server—Designate one machine as the network server. (This machine may also have access to machines beyond the local network.)

  Each machine's domain table only needs to include **badhosts** and **badusers** channels that contain the fully qualified host name of the server. The domain table can be the same for each machine in the network. The network can grow, and machines can be added and removed with no effect on the domain tables of other computers. The disadvantage is that the server receives a great deal of traffic and should be dedicated to its task. In addition, if the server is down, so is all mail between machines. The best policy is a system of machines grouped around a server that has knowledge of another server; each server has knowledge of yet another and so on.

- Distributed Processing—Give each machine's domain table knowledge of each other machine in the network.

  The advantage is that networked machines can operate independently of each other. One machine's crash has no effect on the mail capability of the others (unless that machine is connected to another network via UUCP). The disadvantages are that system administration gets geometrically more difficult as you add or remove machines to or from the network. When a machine or user is added to or removed from the network, you must update all domain or alias tables to recognize the change. Because a domain table can contain redundant information about the local machine, you can use one domain table on every machine. You should only use distributed processing for small networks.

  You can greatly simplify the distributed processing configuration by running a name server program on your network. For details on setting up the name server, see Chapter 6, "Configuring the Domain Name Service" in the *Networking Guide* and "Selecting name server lookup method" (page 72).

- Gateways—Use a gateway to connect a network to another local area or wide area network. (This setup is derived from the server setup.) In this case, the server machine is typically connected to more than one network.

  In addition to containing the names of the local machines on the LAN, a gateway machine also contains the names of the other machines to reach over the other networks. This information is kept in the respective domain and channel tables for the other networks and also in the *root.dom* table on the gateway machine. Other machines on the LAN use the **badhosts** channel to route non-local mail to the gateway machine, or a *root.dom* table that lists all the top-level domains as routing through the gateway. To avoid overloading the gateway machine, the other machines on the LAN should use local domain tables as described in the earlier "Distributed Processing" bullet.

Channel tables

MMDF uses the channel tables to determine the channel to use for outgoing mail and the address of the host on that channel. Channel tables map the fully qualified host name (as determined from the domain table (page 111) entries) to channel-specific addressing information. For example, the UUCP channel table maps host names to UUCP paths (using exclamation points) specifying how to get to each host. Channel tables are created by default in the */usr/mmdf/table* directory (or the directory specified by the **MTBLDIR** parameter in the *mmdftailor* file.)

SCO systems include two channel tables in */usr/mmdf/table*:

Channel table	Channel	Describes
local.chn	local	local machine
list.chn	list	list-processor

The local.chn table

The *local.chn* table contains entries describing all the names local host is called, mapping them to the local host name. For example, if the local host is *volga.mynet.com*, then the local host is known as *volga* on the local machine. The *local.chn* table maps the local host name (on the right) to the different ways people might refer to *volga*. This file looks like this:

```
volga:          volga
mynet.COM:      volga
volga.mynet.COM:        volga
```

This table should minimally include entries for the local host name and the full local domain (the values of the *mmdftailor*(F) **MLNAME** and **MLDOMAIN** parameters).

The list.chn table

The *list.chn* table contains information about the **list-processor** program:

```
list-processor: list-processor
list-proc:      list-processor
```

These entries tell MMDF to pass mail addressed to a mailing list to the **list-processor** program. See "Alias and mailing list tables" (page 116) for more information.

The uucp.chn table

The *uucp.chn* table is created if UUCP is configured on your system and if you add any hosts (page 80) that your host will connect to using the UUCP channel. The table describes how to route mail to those hosts. For example, the format of this file looks like:

```
mcvax.UUCP:     uunet!mcvax!%s
sri-nic.ARPA:   uunet!sri-nic.arpa!%s
uunet.uu.NET:   uunet!%s
```

The left column contains the UUCP host name from the domain tables; the right column describes the UUCP address that MMDF uses to direct mail to that host. The "%s" at the end of the UUCP address means to use the rest of the address from this point on. In other words, when mail is addressed to the user *hillis* at *mcvax.uucp*, the UUCP channel passes the mail to UUnet along with the rest of the UUCP address (*mcvax!hillis*). The second entry in this example shows how a domain name (*sri-nic.arpa*) can be used within a UUCP path.

Channel table entries for the UUCP channel (in the *uucp.chn* file) when the destination machine is multiple hops away, appear like this:

```
stooges.UUCP:   moe!curly!larry!stooges!%s
```

Specify the ***address*** of the host on the right hand side, where the ***address*** is a UUCP path.

For more information about configuring UUCP, see Chapter 7, "Connecting to other computers with UUCP" in the *System Administration Guide.*

The smtp.chn table

The *smtp.chn* table is created if TCP/IP is configured on your system and if you add any hosts (page 80) that your host will connect to using the SMTP channel over TCP/IP. The table describes how to route mail to those hosts and includes their IP addresses. The format of this file looks like:

```
yangtze.mynet.COM:      192.0.0.1
mekong.mynet.COM:       192.0.0.2
seine.mynet.COM:        192.0.0.3
```

The left column contains the fully qualified host names for the hosts that you connect to with TCP/IP and the right column contains the IP addresses. For more information about configuring MMDF to route mail over TCP/IP, see Chapter 3, "Administering TCP/IP" in the *Networking Guide.*

Alias and mailing list tables

System-wide alias and mailing list database tables are created by default in the */usr/mmdf/table* directory (or the directory specified by the **MTBLDIR** parameter in the *mmdftailor* file.)

SCO systems include one channel table called *alias.n* in */usr/mmdf/table*.

Three common types of alias definitions include:

- system administrative aliases for the local host, for delivery of mail to accounts such as *root*, *mmdf*, *postmaster*, and *uucp*. Each host should minimally have an alias defined for *postmaster* (page 74).
- aliases for mailing lists. With list aliases, error messages and mail about problems with the list are sent to the list owner, not the sender. See "Maintaining mailing lists" (page 84).
- aliases mapping users to their "home" host machines (the machines on which they receive their mail). This is used especially if you hide host names (page 71) at your site. See "Associating users with a home machine" (page 85).

See also:

- the *mmdftailor*(F) manual page

How alias tables are created

When you use the **Alias Administration Manager** (page 82) to add an alias, you specify whether you want it to be a public address (page 83) and whether you want to allow bypass (page 83). In addition, you can use an alias to redirect or pipe mail (page 84) to another file or program. Your specifications when defining the alias determine which alias table will contain the alias entry.

These alias tables may be created:

Alias table	Alias description
alias	allow bypass, not public, not redirected
alias.n	no bypass, not public, not redirected
alias.p	allow bypass, public, not redirected
alias.t	no bypass, not public, redirected or piped
alias.np	no bypass, public, not redirected
alias.nt	no bypass, not public, redirected or piped
alias.pt	allow bypass, public, redirected or piped
alias.npt	no bypass, public, redirected or piped

How MMDF uses alias tables

When mail is addressed to *postmaster*, MMDF routes the mail by first searching the alias tables to expand the *postmaster* alias to the associated user name. For example, the *postmaster* entry in the alias table:

```
postmaster:     david
```

Each time MMDF expands an alias successfully, it goes back to the first alias table and begins another search. In a later search, it might find another alias associating *david* (from the example above) with a local machine name.

MMDF then uses this information when searching the various domain (*.dom*) tables (page 111), which map the local machine name to a fully qualified host name, and the channel (*.chn*) tables (page 115), which map the fully qualified host name to information on how to route the message. "How MMDF routes mail" (page 107) further explains the process for searching configuration files.

Assigning Mail IDs

Normally a user's login name is used as identification in a mail address. As an alternative, you can disassociate login names from mail addresses by assigning a Mail ID for each user. To do so, define the **MMAILID** parameter in the */usr/mmdf/mmdftailor* file to be 1. See "Editing MMDF configuration files manually" (page 120).

Then map Mail IDs to login names in two table files: *users* and *mailids*. The location of these table files is defined by **MTBLDIR** in *mmdftailor*; the default is */usr/mmdf/table*. Each table has lines with two entries per line (user and Mail ID, or Mail ID and user). The *users* file is used to map user names (login ids) to Mail IDs, for example:

```
rogerr  Roger
```

The *mailids* file is used to map Mail IDs to user names, for example:

```
Roger rogerr
```

A user can have more than one entry in the *mailids* file, but should have only one entry in the *users* file (the *users* file defines default Mail IDs).

NOTE If the use of Mail IDs is enabled, users must have an entry in the *mailids* file before they can send or receive mail.

The mmdftailor file

The */usr/mmdf/mmdftailor* file defines all the MMDF mail attributes for the local machine, such as its host name, the domain, channel, and alias tables to use, as well as how to set up each channel, and how to perform logging.

MMDF is distributed with a simple *mmdftailor* file that is configured for local mail only. When you perform the initial MMDF configuration, the **MMDF Configuration Manager** modifies the organization of the default *mmdftailor* file. When you make changes using any of the MMDF administration managers or the **MMDF Configuration Manager**, modifications are made to this and other MMDF configuration files.

You can modify *mmdftailor* and other configuration files with a text editor (page 120). However, we recommend that you use the MMDF administration managers and the **MMDF Configuration Manager** whenever possible. When you do make changes with a text editor, be sure to rebuild the hashed database (page 121) when you are finished.

This section describes some of the common MMDF keywords in *mmdftailor*. For a complete list, see the *mmdftailor*(F) manual page.

AUTHLOG controls authorization logging information. Example format:

```
AUTHLOG /tmp/mmdf/mmdfauth.log, level=FST, size=40, stat=some
```

The **AUTHLOG** level must be at least FST, or MMDF does not save any authorization logging information. See "Specifying channel authorization levels" (page 128) and the section on **MCHANLOG** in the *mmdftailor*(F) manual page.

MFAILTIME is the time (in hours) a message can remain in a queue before MMDF sends a failed mail message to the sender and purges the message from the queue. Example format:

```
MFAILTIME       168
```

MLCKTYPE specifies the locking protocol MMDF uses when locking users' mailboxes. This is useful if the users on the system use third-party MUA's that use a lock file that is different from the standard UNIX System V lock file. Example format:

```
MLCKTYPE advisory
```

MMDF locking keywords	
advisory	System V fcntl( ) kernel locking protocols.
v7	Version 7, System V Release 3, and earlier locking protocols. Creates a file called ***username.lock*** in */usr/spool/mail*. ***username*** is the name of the user's mailbox.
xenix	XENIX® system locking protocols. Creates a file called */tmp/**username**.mlk*. ***username*** is the name of the user's mailbox.
all	All of the above locking protocols (default).

The default locking protocol is **all**; however the **MMDF Configuration Manager** sets **MLCKTYPE** to **advisory**.

If you specify more than one locking protocol on the **MLCKTYPE** line, MMDF must satisfy all the locking protocols before a mailbox is considered locked. For example:

```
MLCKTYPE advisory, xenix
```

In this case, MMDF must lock the mailbox, using the **fcntl**() kernel file locking protocol *and* create a file called */usr/spool/mail/**username**.lock*. If it fails to perform both of these locks, MMDF releases the successful lock and tries again later.

MMAXHOPS specifies the maximum number of "Received:" or "Via:" lines a message can contain before the MMDF considers that the message is looping and rejects it. Example format:

```
MMAXHOPS        20
```

MWARNTIME specifies the time (in hours) that a message can remain in a queue before MMDF sends a warning message about delayed delivery to the sender. Example format:

```
MWARNTIME       72
```

MMSGLOG controls the logging information produced by the **deliver**(ADM) and **submit**(ADM) programs. Example format:

```
MMSGLOG /tmp/mmdf/mmdfmsg.log, level=FST, size=40, stat=some
```

See the *logs*(F) manual page and the section on **MCHANLOG** in the *mmdftailor*(F) manual page.

MCHANLOG controls MMDF logging, except for information controlled by **AUTHLOG** and **MMSGLOG**. See the *mmdftailor*(F) manual page for details.

MSLEEP specifies the number of seconds that the **deliver** daemon sleeps between scanning the queues. The default is 600 (10 minutes); we recommend 60 seconds. Example format:

```
MSLEEP    60
```

Editing MMDF configuration files manually

If you modify your MMDF configuration files manually (using a text editor), keep in mind these guidelines:

- Always perform configuration while logged in as *mmdf*.
- Always rebuild the hashed database (page 121) after modifying files.

- If you make any changes to the channel configuration (to, for example, add or remove a channel), restart the **deliver**(ADM) daemon (page 98).

See also:

- the **dbmbuild**(ADM) manual page

Rebuilding the hashed database

MMDF maintains configuration information in a hashed database for use in quickly verifying addresses and efficiently assigning channels to submitted messages. The **MMDF Configuration Manager** and the MMDF administration managers rebuild the MMDF hashed database automatically after modifying the configuration files.

However, if you modify */usr/mmdf/mmdftailor* or the files in the */usr/mmdf/table* directory **after** running the **MMDF Configuration Manager**, you must rebuild this database manually each time you make a modification.

To do this, enter these commands as user *mmdf*:

```
cd /usr/mmdf/table
./dbmbuild
```

NOTE If you specify any options on the **dbmbuild**(ADM) command line, you must add the **-n** option as well. This option creates a new database instead of updating the existing one. To watch the progress of the build, enter **dbmbuild -vn**.

Mail headers

For a mail header to be correct, it must include these three lines in this format:

```
From: sender
To: recipient
Date: Weekday DD Mon YY hh:mm:ss
```

Here is an example:

```
From: fred@mynet.COM (Fred Astaire)
To: ginger
Date: Wed 03 Apr 91 12:21:23 PST
```

In addition, most MUAs and mail submission programs such as **mail**(C) and **scomail**(XC) add extra header lines. For example, the MUA might add the "Message-Id" header. If the MUA does not add this header, MMDF adds it, as well as any "Received" headers. The user can also add other headers. For example, if the user specifies a carbon-copy recipient, the message header includes a "Cc:" line. MMDF allows these additional header lines, but does not use them; the recipient mail server and MUA handle these header lines.

In the example above, note the format of the address in the "From:" line. This is an example of a DOMAIN NAME. MMDF uses the domain name to determine how to route the message.

Mailbox locking

MMDF locks users' mailboxes while they are being modified. By default MMDF is configured to use UNIX System V **fcntl()** kernel locking protocols. You can choose to use XENIX system locking protocols, v7 protocols (Version 7 file-based locking), or all protocols. If MMDF fails to perform the specified locks, it releases any lock it has placed and tries again later.

See the section on the **MLCKTYPE** keyword in "The mmdftailor file" (page 118).

Specifying MMDF authorizations

Control the flow of mail through your host using MMDF authorizations. You can authorize mail to be sent from your host on specific channels, based on the sending or receiving host, the user who originates the mail, or both. You can also authorize mail routed through your domain.

Host-based authorizations
: This form of authorization control (see "Specifying host-based authorization" (page 123) for instructions) is often used to ensure that hosts that are not part of the site's private network do not use that network to send mail to other parts of the world. For example, you can use host-based authorization to allow mail to enter your domain, but restrict the mail passing through it.

User-based authorization
: Control the flow of mail both to and from your host and network using this form of authorization control (see "Specifying user-based authorization" (page 125) for instructions). For example, if your site is on the Internet, you might want to allow your employees (but no one else) to send mail from their home machines through your system to the Internet. In this case, you would want to set up MMDF to authorize mail from employee users to pass through your machine, but prevent mail from other users from passing through.

Routing-based authorization
: Keep mail that enters your domain from leaving your domain using this form of authorization control (see "Setting routing-based authorization" (page 126) for instructions). Users in your domain can send mail to people both inside and outside the domain, but users outside the domain cannot route mail through your domain to another destination. For example, if your domain includes machines in different cities with links to different

outside computers, people from outside the domain might use these links to send mail from one city to another, through your domain. In this case, you can set up MMDF authorization to prevent people from using your system to transfer mail.

See also:

- "Specifying both host and user authorization" (page 127)
- "Specifying channel authorization levels" (page 128)
- "Changing error logging levels" (page 128)

Specifying host-based authorization

MMDF provides authorization checks on the next "hop" host in the address route. If someone specifies a route through your host, MMDF on your host can authorize the next hop in that route. In other words, MMDF does not perform authorization based on the final destination unless the final destination is the next hop.

To control authorization on a per-host basis:

1. Log in as *mmdf* and declare a table in the file */usr/mmdf/mmdftailor* using the **MTBL** keyword. The table will specify the privileges for all hosts that do not belong to the network. For information on editing *mmdftailor*, see "Editing MMDF configuration files manually" (page 120) and the *mmdftailor*(F) manual page.

   This example declares the table *world-auth*:

   ```
   MTBL "world-auth", file="authinfo/world"
   ```

   The **file** parameter specifies that *world-auth* is stored in */usr/mmdf/table/authinfo/world*. You will create the actual table later in this procedure.

2. Define two channels using **MCHN** definitions in */usr/mmdf/mmdftailor*: one channel for the hosts that belong to the network and one for the hosts that do not belong. Use one or more of these parameters to specify how the information in the world-auth table is to be utilized:

   indest — When mail is sent to a particular host or via a particular channel, the table determines whether the source host is allowed to send mail to the destination host or via that channel. Example format: **MCHN indest="world-auth"**

   outdest — When mail is sent to a particular host, the table verifies that the source host or channel is allowed to send mail to the destination host.

insrc When mail arrives from a particular host (the source host), the table verifies that that host is allowed to send mail to the destination host by authorizing either the destination host or the channels used to access the destination host.

outsrc When mail arrives from the source host via a particular channel, the table determines if that source host or channel is allowed to send mail to the destination host.

For example, define a channel (called *localnet* in this example) for all hosts on the local network:

```
MCHN localnet, auth=free, show="LOCALNET Delivery",
        ap=822, mod=imm
```

The "auth=free" authorization setting is the default; you do not have to specify it explicitly as in this example. See "Specifying channel authorization levels" (page 128).

Define a channel (called **world** in this example) for all the hosts not in the *localnet* network:

```
MCHN world, auth=inblock, indest="world-auth",
        show="WORLD Delivery", ap=822, mod=imm
```

In this case, anyone can send mail out on the **world** channel, but MMDF checks the **world-auth** table to authorize the destination of mail arriving on this channel.

3. Create a channel table file in */usr/mmdf/table* for each of the channels you just created. In the above example, you would create the files *world.chn* and *localnet.chn*. In those files, include descriptions of each host accessed via that channel. See "Channel tables" (page 115) for more information.

4. Create the **world-auth** table (the file */usr/mmdf/table/authinfo/world* in our example). Include lines like these:

```
world:
local:
localnet:       moocow.uucp
```

The entries on the left side of the authorization table specify that if the destination host for a message is on either the **world** or **local** channels, MMDF authorizes anyone using the **world** channel as an input channel to send mail.

The entry on the right side of the **localnet** channel entry specifies the hosts and channels that are authorized to send outgoing mail using the **localnet** channel. In this case, *moocow.uucp* is the only machine allowed to pass mail into the *localnet* network.

5. Rebuild the hashed database using the information in "Rebuilding the hashed database" (page 121).

Specifying user-based authorization

To configure MMDF authorization on a per-user basis, you must first set up the authorization level on the channels that you want to restrict and create appropriate channel tables, as described in "Specifying host-based authorization" (page 123).

To set up authorizations for specific users:

1. Log in as *mmdf* and declare the user authorization table in the file */usr/mmdf/mmdftailor* using the **MTBL** keyword. For information on editing *mmdftailor*, see "Editing MMDF configuration files manually" (page 120) and the *mmdftailor*(F) manual page.

   For example, if the name of file containing the user authorization table is *auth.user*:

   ```
   MTBL auth, file="auth.user", show "Per-user authorization"
   ```

   You must call the per-user authorization table *auth*; MMDF treats any table called *auth* as the per-user authorization table.

2. Create the user authorization table in */usr/mmdf/table* (in the example in step 1, the file would be */usr/mmdf/table/auth.user*). Use this format for the table contents:

   ***username*: *keyword channel* [, *channel*]**

   The ***username*** can be a local or remote user name, ***keyword*** describes the actions that you can authorize users to perform, and ***channel*** is the channel name on which the authorizations apply. The ***keywords*** are:

both	allows user to send and receive mail
send	allows user to send mail only
recv	allows user to receive mail only
expire	expires access privileges for the user (and includes this information in any error mail)

   MMDF treats any other ***keyword*** as **expire**, except that MMDF sends the text of the action instead of "expire" to the user in error mail.

   Include one line in the table for each user to whom you want to grant mail access. Any users not listed are not authorized to use any channel except the channels set to the **free**, **inlog**, or **outlog** authorization levels (see "Specifying channel authorization levels" (page 128)).

   Example I:
   To set up access authorizations for a local user, specify the unqualified user name. For example, the local user *andrei* can both send and receive mail on the SMTP and UUCP channels:

   ```
   andrei: both smtp,uucp
   ```

However, if you set up host-based authorization to restrict access to one of these channels, for example UUCP, *andrei* might not be authorized to send or receive mail on that channel.

Because mail on the local channel is not restricted, *andrei* can pass mail through this channel even though the user authorization list does not include "local" in the list of channels.

Example II:
To set up access authorization for a remote user, specify the fully qualified address of that user. For example, to allow *melissa* on the machine *silly.org* to send mail through TCP/IP (the SMTP channel) on this host, add a line like this to the authorization table:

```
melissa@silly.org: send smtp
```

If mail arrives for *melissa* through UUCP, or if she tries to send mail through the UUCP channel, MMDF rejects the mail.

Example III:
To expire a particular user's access and tell MMDF to send an error message, add the message to the user authorization table line for that user. For example, to expire *aaron@thames.com*'s access and send the text "No more mail for you!", include a line like this one:

```
aaron@thames.COM: "No more mail for you!" uucp
```

3. Rebuild the hashed database using the instructions in "Rebuilding the hashed database" (page 121).

Setting routing-based authorization

To set up routing-based authorization for hosts that are not in your domain (*mynet.com* in this example):

1. Log in as *mmdf* and declare an authorization table in the file */usr/mmdf/mmdftailor* using the **MTBL** keyword. For information on editing *mmdftailor*, see "Editing MMDF configuration files manually" (page 120) and the *mmdftailor*(F) manual page.

   For example:

   ```
   MTBL "world-auth", file="authinfo/world"
   ```

   This declares a table called **world-auth** that is maintained in the file *authinfo/world*. This table will contain the authorization information for the **world** channel.

2. Specify a channel for your domain. For example, for a channel called *mynetwork*, create an **MCHN** entry like this:

   ```
   MCHN mynetwork, auth=free, show="MYNET Network Delivery",
           ap=822, mod=imm
   ```

3. Define a channel for the rest of the hosts that are not in the local domain (again, this appears as one line in *mmdftailor*):

```
MCHN world, auth=inblock, auth=dho, indest="world-auth",
        show="WORLD Delivery", ap=822, mod=imm
```

   The "auth=indest" parameter specifies that when **world** is the input channel, MMDF checks the *authinfo/world* file to verify that the inbound host is authorized to send mail to the destination. See "Specifying channel authorization levels" (page 128).

   When you specify the "auth=dho" parameter on a channel, MMDF replaces the "host" (in host-based authorization) used to check authorization with a route. The route is either from the source or to the destination, depending on which "auth" level that you specify. MMDF replaces the local section of the route (the user's name) with the string "username". Then, MMDF compares this route to the entries in the table, to determine if the message is authorized or not.

4. Create a channel table file in */usr/mmdf/table* for each of the channels you just created. In the above example, you would create the files *mynetwork.chn* and *world.chn*. In those files, include descriptions of each host accessed via that channel. See "Channel tables" (page 115) for more information.

5. Create the *authinfo/world* file, and include entries like these:

```
world:
username@mynet.com:
username@larry.mynet.com:
username@moe.mynet.com:
username@curly.mynet.com:
```

   This table authorizes MMDF to deliver any mail addressed to people in the *mynet.com* domain arriving or leaving on the **world** channel. This does not allow mail to pass through the **mynetwork** channel to a destination outside the *mynet.com* domain.

6. Rebuild the hashed database with **dbmbuild**.

Specifying both host and user authorization

The default authorizations allow MMDF to deliver a message if either the user-based (page 125) or host-based (page 123) authorization tables allow it. Thus, if a user table authorizes a user to send mail over a channel, but the host table does not, that user can still use the channel (and vice versa).

To require that both the user and host tables authorize access to a channel, specify **auth=hau** in the **MCHN** definition in */usr/mmdf/mmdftailor*.

Specifying channel authorization levels

To control authorization on a particular channel, use the **auth** keyword to set an authorization level on the **MCHN** line in the */usr/mmdf/mmdftailor* file.

MMDF provides seven levels of authorization:

free Performs no authorization checks (default). Any channel that you do not set authorization for is set to **free**.

inlog Logs incoming authorized and unauthorized access, but allows mail to pass.

outlog Logs outgoing authorized and unauthorized access, but allows mail to pass.

inwarn Logs incoming unauthorized access, transfers the message, and sends a warning to the originator.

outwarn Logs outgoing unauthorized access, transfers the message, and sends a warning to the originator.

inblock Logs the incoming unauthorized access attempt and bounces the mail.

outblock Logs the outgoing unauthorized access attempt and bounces the mail.

Example format:

```
MCHN auth=inblock
```

Changing error logging levels

You can have MMDF maintain a complete log of authorization attempts (page 129) with reasons for failure or success.

By default, the **MMDF Configuration Manager** sets up MMDF to log fatal errors only. To change the logging level, start the **Channel Administration Manager** (page 85). Then:

1. Highlight the channel to modify.
2. Select **Modify** from the **Channel** menu.
3. Click on **Log File**.

4. Use the default log file or specify a different log file by clicking next to **Select** and typing the name in the "File" field.
5. Select the Log level:

fatal errors only	Logs only fatal errors. Produces a complete log of authorization attempts.
temporary errors	Logs both temporary and fatal errors. Produces a complete log of authorization attempts.
interesting diagnostics	Logs generally interesting diagnostics and errors. Produces a complete log of authorization attempts.
basic statistics	Logs some basic statistics about movement of messages. Produces a complete log of authorization attempts.
full statistics	Logs full statistics.
program trace listing	Shows a program trace listing of channel activity.
detailed trace listing	Shows a more detailed program trace listing.
every diagnostic possible	Logs all possible diagnostics.

Each level listed here logs the preceding levels. For example, Interesting diagnostics logs Fatal errors and Temporary errors as well. The last three levels are most often used by experienced administrators for analysis of MMDF configuration.

The log file's default maximum size is 1000 blocks. This value, and other log file parameters, can be modified; see the description of **MCHANLOG** in *mmdftailor*(F).

See also:

- "Specifying MMDF authorizations" (page 122)
- the *logs*(F) manual page
- "Channel tables" (page 115)

Authorization log files

MMDF authorization log files contain descriptions of authorization attempts and reasons for failure or success. Log files are produced by defining the appropriate authorization logging level (page 128).

The format of messages in the authorization log file is similar to other MMDF log files. Each message includes the date, time, message source, and message ID, followed by the log-specific information. The message also includes an "end of processing" message that describes the message sender and size. Each authorization message can include either one or two reasons for authorizing a particular message.

Message example:

```
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='ucscc'
      a='lisa@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='ucscc'
      a='jane@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:55 AU-0000:  msg.a000561: END size='2102', sender='robert'
```

NOTE Authorization log file messages appear on one line; the examples in this section split the lines for readability.

Authorization message keys

Keys used in authorization log file messages:

i input channel

o output channel

a destination address

r reason for authorization

hi inbound host

ho outbound host

Single-reason authorization codes

If the authorization message includes a single reason for authorization, the "r" key specifies a single authorization code that describes both the inbound and the outbound authorization when you use host-based authorization.

Single-reason authorization codes:

OH outbound host/route

HC outbound host/route and inbound channel

HH inbound host/route and outbound host/route

CC inbound channel and outbound channel

CH inbound channel and outbound host/route

IH inbound host/route

This example uses the **CH** authorization code:

```
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='peaks'
      a='bob@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='peaks'
      a='mike@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:55 AU-0000:  msg.a000561: END size='2102', sender='cooper'
```

In this example, the authorized message has two recipients (*bob* and *mike*). The first authorization message shows that the inbound channel ("i") is the local channel and the outbound channel is **peaks**. The "a" key indicates that the recipient's address is *bob@rsre.ac.uk*.

The reason ("r") given for authorizing the message is **CH**; in other words, the inbound channel (**local**) has authorization to send mail to the given outbound host or route (specified by "ho"), in this case *username@rsre.ac.uk*.

Two-reason authorization codes

Two-reason authorization codes describe the reason for authorization in terms of user-based authorization:

IL inbound channel, outbound = **LIST**

OL outbound channel, dest = **LIST**

IS inbound channel by sender

OS outbound channel by sender

IR inbound channel by receiver

OR outbound channel by receiver

I inbound channel, log unauthorized access

O outbound channel, log unauthorized access

NOTE MMDF only uses these authorization codes when you set the "auth=inlog" or "auth=outlog" parameters of the AUTHLOG keyword (page 128) in */usr/mmdf/mmdftailor*.

The message in this example uses two-reason authorization (if no authorization is required for a channel, MMDF leaves the reason field ("r") empty):

```
4/29  9:53:09 AU-0000:  msg.a000653: i='local' o='mynet'
      a='john@edxa.ac.uk' r='' r='OS'
4/29  9:53:10 AU-0000:  msg.a000653: END size='197', sender='david'
```

In this example, the message arrived (with no authorization required) on the local channel and is authorized to leave on the **mynet** channel because the sender (*david*) is authorized to use it (**OS**).

See also:

- the *mmdftailor*(F) manual page

Troubleshooting MMDF

MMDF is configured for the local system by default. Therefore, users should not experience problems with local mail. Generally, when mail does not work, there is either a problem with the network, or with MMDF configuration.

When mail problems occur, first verify that any network connections work correctly (for example, UUCP or TCP/IP connections). Make sure you can **ping**(ADMN) each of the machines and that you can **telnet**(TC) and/or **rlogin**(TC) to the remote system. See the *Networking Guide* for more information on troubleshooting networks.

If you are configuring MMDF over a UUCP connection, make sure you can use **uucp**(C) to transfer files between systems. If you are connecting to UUCP with a modem, verify that the modem is installed and configured correctly. See "Troubleshooting modems" in the *SCO OpenServer Handbook*.

If networks are working and you still experience problems with **mail**, the problem is most likely with the MMDF configuration or the **deliver** daemon. This section discusses some common problems and solutions.

If you need to call SCO Support, be sure to have the following files ready when speaking with an SCO Support engineer:

/usr/mmdf/mmdftailor
/usr/mmdf/table/.chn*
/usr/mmdf/table/.dom*
/usr/mmdf/log/.log*

We also recommend that you be in front of the machine having the problem when speaking to an SCO Support engineer, so you can try suggestions we may have.

Failed mail error

If the system returns your mail immediately in a mail message with the words "Failed Mail" in the "Subject" line, there is a problem with the system channel or domain tables. Use the following procedure to check this:

1. Log in as *mmdf*.
2. Change directory to */usr/mmdf/bin*.
3. Invoke **checkaddr**(ADM) with the mail address that you used. For example, if **mail sylvia@seattle** failed, enter:

   ./checkaddr sylvia@seattle

4. If **checkaddr** displays "Unknown Host Domain," either the host or domain is not in the channel or domain files, or the */usr/mmdf/mmdftailor* file contains an error.

   If **checkaddr** displays "OK," then the host and domain are in the channel and domain files, but are incorrectly configured. Check the address with **checkaddr -w** to diagnose this problem.

5. Verify that your domain (page 111), channel (page 115), and */usr/mmdf/mmdftailor* (page 118) files are set up correctly.

Mail does not work, no returned mail

If mail does not reach its destination but MMDF does not return unsent mail, either **deliver**(ADM) is not running properly, or the **mmdftailor**(F) file contains an error. See "Customizing mail delivery" (page 98) for information about the **deliver** daemon.

Check to see that **deliver** is running using **ps -ef**. If **deliver** is not running, use this procedure:

1. Log in as *mmdf*.
2. Start the **deliver** daemon manually with the following commands:

   /usr/mmdf/bin/daemon stop
   /usr/mmdf/bin/daemon start

   This starts **deliver** sweeping the mail queues in the background every 60 seconds. For more information, see the **deliver**(ADM) manual page.

3. If running **deliver** manually does not work, verify that your domain (page 111), channel (page 115), and */usr/mmdf/mmdftailor* (page 118) files are set up correctly.

Unable to reply to mail from a Micnet host

When a host is running MMDF and is connected to a non-MMDF computer via Micnet, mail users on the MMDF machine are unable to reply to mail originated on the non-MMDF machine.

To allow mail replies between an MMDF and non-MMDF systems via Micnet, a global alias file must be maintained on each machine. Use **netutil**(ADM) to create a global alias file on the non-MMDF machine and to distribute the file to MMDF machines. On the MMDF machines, use **mmdfalias**(ADM) to convert the global alias file to MMDF style:

1. Copy the non-MMDF global alias file to */usr/lib/mail/aliases*.
2. Log in as *mmdf* and change to the */usr/mmdf/table* directory.
3. Run the commands:

   /usr/mmdf/table/tools/mmdfalias
   dbmbuild

NOTE Any modifications to the global alias file must be made using **netutil** and distributed again.

Mail command hangs

If the **mail**(C) command hangs when a user tries to read mail, check the following:

1. Verify that the user's mailbox exists in the */usr/spool/mail* directory. Unless your system was configured differently as described in "Specifying the location of user mailboxes" (page 73), the */usr/spool/mail* directory should contain a file for each user who receives mail.
2. Make sure that the owner and group ID of the */usr/spool/mail* directory is *mmdf*.
3. Make sure that the mailbox is not too large (1000 messages is regarded as too large).
4. Check for these lock files:

   */usr/spool/mail/**user_name**.lock*
   */tmp/**user_name**.mlk*

   user_name is the user's account name. If a lock file is found, remove it with this command:

   /usr/mmdf/bin/cleanlck

Mailbox locking problems

You may experience mailbox locking problems, signified by these error messages:

- `mail: Cannot read lock file '/usr/spool/mail/username'`

  Or, if mail is delivered to $HOME/.mailbox, the message is

  ```
  mail: Cannot read lock file 'username/.mailbox'
  ```

- This error message can be found in */usr/mmdf/log/msg.log* and *chan.log*:

  ```
  10/ 5 10:15:06 local 0913:  [SYSERR (13) Permission denied]
                              open of /usr/spool/mail/root

  10/ 5 10:15:06 local 0913:  [SYSERR (13) Permission denied]
                              can't open mailbox '/usr/spool/mail/root'
  ```

The MMDF configuration sets **MLCKTYPE** to **all** (**MLCKTYPE** allows you to specify the locking protocol for MMDF to use when locking users' mailboxes).

The **MLCKTYPE** parameter is useful if the users on the system use third party mail user agents (MUAs) that use a lock file that is different from the standard System V lock file.

This mailbox locking problem occurs when MMDF fails to unlink the */usr/spool/mail/username.lock* or */tmp/username.mlk* file, which is created to lock the mailbox. If mail is delivered to $HOME/*.mailbox*, the lock file is */tmp/.mailbox.lock*. A subsequent **fcntl**(M) (kernel file locking) process returns with a permission denied error because it cannot lock the mailbox.

To resolve the locking problem, log in as *root* and enter this at the *root* prompt:

/usr/mmdf/bin/cleanlck

NOTE You can use **MLCKTYPE** (in */usr/mmdf/mmdftailor*) to specify other locking protocols. See "The mmdftailor file" (page 118). However, changing locking protocols is not recommended.

Undelivered messages in /usr/spool/mmdf/lock/home

If **cron**(C) queues a file with this error message in */usr/spool/mmdf/lock/home/q.local* to be delivered to the *sys* user:

```
To:sys

cron: can't change directory to your home directory.
Your commands will not be executed.
```

the problem may be that the sys home directory was removed accidentally.

Other symptoms of this problem are these errors:

- `Out of inodes on dev (1/40)`
- `Out of space on dev (1/40)`
- `no more processes`
- `no file`

To resolve this problem:

1. Recreate the */usr/sys* directory by logging in as *root* and entering these lines:

   mkdir /usr/sys
   chmod 755 /usr/sys
   chown sys /usr/sys
   chgrp sys /usr/sys

2. If your disk filled up due to the problem, recover disk space by entering:

   cd /usr/spool/mmdf/lock/home

   and removing the messages in the *addr*, *msg*, and *q.local* directories.

   Clean the log files by entering:

   > /usr/spool/mail/sys
   > /usr/spool/mail/adm
   > /usr/spool/mail/root
   > /usr/mmdf/log/msg.log
   > /usr/mmdf/log/chan.log
   > /usr/adm/messages
   > /etc/wtmp

3. Boot the system. Enter the *root* password to bring up the system in single user mode when you see this prompt:

   ```
   Type <ctrl>-d to proceed with normal startup,
   (or give root password for System Maintenance)
   ```

   At the prompt, enter:

   /etc/fsck -s /dev/root

   This file system check with an **s** option updates the free list of available inodes and blocks in the superblock.

The undelivered messages problem should now be resolved.

Using name server resource records with MMDF

Name service can be used to route mail over TCP/IP (SMTP) **if you have TCP/IP installed and configured**. See "Selecting name server lookup method" (page 72).

MMDF can use these name server resource records: **A**, **CNAME**, **MB**, **MR**, **MG**, and **MX**. See "Mail-related name server resource records" (this page).

Use **MX** resource records by defining a channel (page 85) that uses a table that is configured to query a name server. See "Table Data" in "Modifying MMDF table parameters" (page 110).

Use **A** and **CNAME** resource records by including a domain (page 94) that uses a table that is configured to do subdomain lookups. See "Lookup Options" in "Modifying MMDF table parameters" (page 110). Usually the root domain is used for this purpose.

Use **MB**, **MR**, and **MG** resource records by including an **ALIAS** declaration in the */usr/mmdf/mmdftailor* file. For example:

```
MTBL name=nsalias, flags=ns, show="DNS MR, MB records.
        flags=alias
ALIAS   table=nsalias
```

The program **host**(ADMN) may be used to exercise the interface routines to the name server that are used by MMDF.

See also:

- the **named**(ADMN) manual page
- "Editing DNS resource records" in the *Networking Guide*

Mail-related name server resource records

The **MB**, **MR**, **MG**, and **MX** name server resource records are explained here:

MX records Mail exchanger records. **MX** resource records indicate that mail directed to a given domain may be routed via this host. They have the format:

```
domain  IN      MX      preference      mail.exchanger
```

MB records Mail box records. **MB** resource records indicate that the given user's mail box is on the specified host. They have the format:

```
user    IN      MB      host
```

MMDF translates this to ***user@host***, by appending ***@host*** to ***user***.

MR records Mail rename records. They have the format:

```
user    IN      MR      new.mail.address
```

MMDF replaces ***user*** with ***new.mail.address***. It is an alias in the usual sense of the word.

MG records Mail group records. They have the format:

```
list    IN      MG      user1@host1
list    IN      MG      user2@host2
```

MMDF uses **MG** records to build a list of mail addresses to send the mail to.

NOTE The example records shown here are simplified for clarity and do not correspond exactly to the **named**(ADMN) syntax.

See also:

- the **named**(ADMN) manual page
- "Editing DNS resource records" in the *Networking Guide*

MMDF delivers mail twice or not at all

If MMDF delivers mail twice or does not deliver it at all, the problem may be with the **deliver**(ADM) program. Mail is delivered immediately by default. If two people attempt to send mail at the same time, two deliver daemons, both of which look at the same mail queue, are started at the same time. If this happens, mail may be delivered two or more times.

Because of a problem with the deliver program, mail coming from a remote site may not be delivered at all.

To solve either of these problems:

1. Log in as *mmdf*.
2. Use the **Channel Administration Manager** to specify Regular delivery mode (page 88) for the channels in use. Note the channels whose configuration you change.

NOTE The concept of "delivery" in this context does not mean that the mail is necessarily put in a mailbox file. It usually means that the mail is transferred from the MMDF spool directories (*/usr/spool/mmdf*) to the appropriate spool directories for the channel being used. For example, if the uucp channel is being used, the mail ends up under */usr/spool/uucp*. Depending on how the channel is configured, transfer of mail to the channel's spool directories may or may not trigger immediate transfer of mail to its destination mailbox.

Smtp cannot establish a remote connection

You can have a working TCP/IP configuration with successful implementation of most TCP/IP functions (**telnet, rlogin, ftp,** and **rcp**) between the local machine and a remote machine, but the */usr/mmdf/chans/smtp* program may be unable to establish a connection with the remote site.

You may see an error message like this in */usr/mmdf/log/chan.log* when you attempt to send mail to the remote site if you have the logging level (page 128) set to **Every diagnostic possible** (**FTR** in the *mmdftailor*(F) file):

```
trying 89.0.0.3
can't get connection (115)
[SYSERR(115) connection refused] a3000.sco.com(59000003) no open
sm_gethostid (a3000.sco.COM)
tb_k2val (smtp.chn, first=0, a3000.sco.COM)
tb_k2val failed
qu_wrply( )
(DHST) 'Remote host unavailable'
qu_wrec( ) (15) "Remote host unavailable"
qu_radr
qu_rrec ( )
(4) "aend"
qu_wrply( )
(DHST) 'no valid addresses'
qu_wrec( ) (51) "no valid addresses"
qu_mend
hd_end ( )
qu_pkend
wend
qu_end
```

When **inetd**(ADMN) is started it reads */etc/inetd.conf* and */etc/services*. It then creates a socket for each server and binds each socket to the port for that server.

inetd listens on multiple ports for incoming connection requests. When it receives a request it spawns the appropriate server.

The above error message indicates that the local **smtp** program did not establish a socket connection. This means that the **inetd** process on the remote machine did not open a socket for *smtp* at port 25 because the smtp entry was omitted in */etc/inetd.conf* and */etc/services*.

To resolve the problem:

1. Log in as *root* on the remote machine.
2. If the smtp entry is missing from */etc/inetd.conf*, edit *inetd.conf* and add this line:

```
smtp  stream  tcp nowait mmdf /usr/mmdf/chans/smtpd smtpd
        /usr/mmdf/chans/smtpsrvr smtp
```

NOTE This long entry is all one line.

3. If the smtp entry is missing from */etc/services*, edit *services* and add this line:

```
smtp      25/tcp      mail
```

The line in */etc/services* consists of service name, port number and protocol type (separated by /), and an optional alias name.

4. Restart **inetd** by entering this when in multiuser mode:

```
ps -ef | grep inetd
```

If you see an output similar to this:

```
root  257 1 0  May 11  ?  0:01 inetd
```

then kill **inetd** by entering:

```
kill -1 257
```

Where *257* is the process id.

If the remote machine (to which mail is being directed) is not an SCO system, see the appropriate documentation for your system.

See also:

- the **inetd**(ADMN) manual page
- the *inetd.conf*(SFF) manual page
- the *services*(SFF) manual page

UUCP channel delays mail delivery

If the MMDF UUCP channel hands off mail to UUCP, but the mail does not get delivered immediately, even though you have configured both MMDF and UUCP to deliver messages immediately, there is an omission in the */usr/mmdf/mmdftailor* file.

The first step to take is to make sure that UUCP is set up to call out immediately. The *Systems* file entry for the UUCP host should have the keyword **Any** in the *schedule* field to call out at any time. For example, if the remote UUCP host is called *oratory*, then the entry for *oratory* in */usr/lib/uucp/Systems* should begin:

```
oratory Any ...
```

You must also set up MMDF to transfer the mail to the UUCP channel immediately. Do this by using the **Channel Administration Manager** to specify Immediate delivery mode (page 88) for the **uucp** channel.

Add this line to */usr/mmdf/mmdftailor*:

```
UUXSTR "uux -"
```

NOTE Edit all MMDF files as user *mmdf*.

Message not deliverable

If you get error messages that a message is not deliverable, there are several possible reasons why, some of which include:

- the message includes incorrect address information
- the recipients may be unknown locally or to your name server
- there may have been a delivery timeout because of problems with a computer or network in the delivery path
- a destination mailer can refuse to accept the message
- errors during processing

Example MMDF configurations over TCP/IP

This section provides an example for configuring MMDF for use over a TCP/IP network, using the Simple Mail Transfer Protocol (SMTP) channel. Before configuring MMDF, follow the instructions in "Running the MMDF Configuration Manager" (page 68); the TCP/IP network must be configured and working between the systems that will exchange mail and the user *mmdf* should have an assigned password. Specify TCP/IP as one of the networks to use for mail.

Setting up MMDF on a TCP/IP network

This section describes configuring MMDF on a network of machines connected with TCP/IP, where each computer can directly contact the others on the network. The example machines are *volga*, *columbia*, and *thames*, and the domain (page 94) is *mynet.COM*. The IP addresses and fully qualified domain names of the example computers are:

132.147.118.1 *volga.mynet.COM*
132.147.118.2 *columbia.mynet.COM*
132.147.118.3 *thames.mynet.COM*

For further information, see Chapter 3, "Administering TCP/IP" in the *Networking Guide*. **Example:**

1. Log in as *mmdf* on *volga*.
2. Check that the domain (page 95) *mynet.COM* exists. Because this is the top-level domain used by *volga*, it will have been added at installation time, but you can use the **Domain Administration Manager** to check, and add it if it does not exist.
3. In the **Configuration Manager**, check that the domain name is not hidden (page 71) by selecting this addressing option:

   "From: ***user@volga.mynet.COM***"

   Host entries (page 79) for the other machines in the local domain, *columbia.mynet.COM* and *thames.mynet.COM*, are added by the **Configuration Manager**.

   When you have verified these conditions, these entries will exist in */usr/mmdf/table/local.dom*:

```
volga:          volga.mynet.COM
columbia:       columbia.mynet.COM
thames:         thames.mynet.COM
```

   These entries will exist in */usr/mmdf/table/root.dom*:

```
volga:                  volga.mynet.COM
volga.mynet.COM:        volga.mynet.COM
```

   These entries will exist in */usr/mmdf/table/smtp.chn*:

```
volga.mynet.COM         132.147.118.1
columbia.mynet.COM      132.147.118.2
thames.mynet.COM        132.147.118.3
```

As a result of the initial mail configuration, these entries will exist in */usr/mmdf/table/local.chn*:

```
volga:                  volga
volga.UUCP:             volga
volga.mynet.COM         volga
```

These entries will exist in */usr/mmdf/mmdftailor* (among others):

```
MLDOMAIN        "mynet.COM"
MLNAME          "volga"

MDMN name="root", dmn="", show="Root Domain", table=rootdom
MDMN name="localdom", dmn="mynet.COM", show="mynet.COM Domain",
     table=localdom

MCHN show="Local delivery", name=local, que=local, tbl=local, pgm=local,
     ap=822, mod=imm
MCHN show="SMTP Delivery", name=smtp, que=smtp, tbl=smtpchn, pgm=smtp,
     ap=822, mod=host, confstr="charset=7bit"
```

4. Repeat steps 1–3 on *columbia*. When you have verified the conditions, these entries will exist in */usr/mmdf/mmdftailor* (the **MCHN** and **MDMN** entries will appear as listed above for *volga*):

```
MLDOMAIN        "mynet.COM"
MLNAME          "columbia"
```

These entries will exist in */usr/mmdf/table/local.dom*:

```
columbia:       columbia.mynet.COM
volga:          volga.mynet.COM
thames:         thames.mynet.COM
```

These entries will exist in */usr/mmdf/table/root.dom*:

```
columbia:               columbia.mynet.COM
columbia.mynet.COM:     columbia.mynet.COM
```

These entries will exist in */usr/mmdf/table/smtp.chn*:

```
columbia.mynet.COM      132.147.118.2
volga.mynet.COM         132.147.118.1
thames.mynet.COM        132.147.118.3
```

These entries will exist in */usr/mmdf/table/local.chn*:

```
columbia:               columbia
columbia.UUCP:          columbia
columbia.mynet.COM      columbia
```

5. Repeat steps 1–3 on *thames*. When you have verified the conditions, these entries will exist in */usr/mmdf/mmdftailor* (the **MCHN** and **MDMN** entries will appear as listed above for *volga*):

```
MLDOMAIN        "mynet.COM"
MLNAME          "thames"
```

These entries will exist in */usr/mmdf/table/local.dom*:

```
thames:         thames.mynet.COM
volga:          volga.mynet.COM
columbia:       columbia.mynet.COM
```

These entries will exist in */usr/mmdf/table/root.dom*:

```
thames:                 thames.mynet.COM
thames.mynet.COM:       thames.mynet.COM
```

These entries will exist in */usr/mmdf/table/smtp.chn*:

```
thames.mynet.COM        132.147.118.3
volga.mynet.COM         132.147.118.1
columbia.mynet.COM      132.147.118.2
```

These entries will exist in */usr/mmdf/table/local.chn*:

```
thames:                 thames
thames.UUCP:            thames
thames.mynet.COM        thames
```

MMDF is now configured to send mail among all three systems. Mail should be addressed using Internet style addresses, such as *root@thames.mynet.COM*.

Setting up MMDF using a mail gateway

This section describes setting up MMDF on a network of machines over a TCP/IP Ethernet, using one machine as the mail gateway. In this case, all machines on the Ethernet will forward all mail to one machine, which will then distribute the mail to the appropriate destination.

The purpose of a mail gateway machine is to provide ease of configuration. Establishing a gateway does not in any way prevent the other machines from communicating directly over the Ethernet with **rlogin**, **telnet**, or other communication programs.

The machine names in this example are *seine*, *yangtze*, and *mekong*. The IP addresses and fully qualified domain names of the example computers are:

132.147.118.4 *seine.mynet.COM*
132.147.118.5 *yangtze.mynet.COM*
132.147.118.6 *mekong.mynet.COM*

The gateway machine is *mekong.mynet.COM*.

This section assumes that all configuration files are in the default, newly installed state—not the state in which they were left after following the steps in "Setting up MMDF on a TCP/IP network" (page 142).

1. Log in as *mmdf* on *seine*.
2. Check that the domain (page 95) *mynet.COM* exists. Because this is the top-level domain used by *seine*, it will have been added at installation time, but you can use the **Domain Administration Manager** (page 94) to check.
3. In the **MMDF Configuration Manager** (page 68) check that the domain name is not hidden (page 71) by choosing the addressing option:

   "From: ***user@seine.mynet.COM***"

4. In the **MMDF Configuration Manager**, specify that you want to forward mail addressed to unknown hosts (page 73) over the **badhosts** channel (page 91) to the smart host (the gateway) *mekong.mynet.COM*.

   Host entries (page 79) for the other machines in the local domain, *yangtze.mynet.COM* and *mekong.mynet.COM*, are added by the **MMDF Configuration Manager**.

   As a result of these steps, these entries will exist in */usr/mmdf/mmdftailor*:

```
MLDOMAIN        "mynet.COM"
MLNAME          "seine"

MDMN name="root", dmn="", show="Root Domain", table=rootdom
MDMN name="localdom", dmn="mynet.COM", show="mynet.COM Domain",
     table=localdom

MCHN show="Local delivery", name=local, que=local, tbl=local, pgm=local,
     ap=822, mod=imm
MCHN show="SMTP Delivery", name=smtp, que=smtp, tbl=smtpchn, pgm=smtp,
     ap=822, mod=host, confstr="charset=7bit"
MCHN show="Smart-host Routing for hosts", name=badhosts, que=badhosts,
     tbl=smtpchn, pgm=smtp, ap=822, mod=host,
     confstr="hostname=seine.mynet.COM", host="mekong.mynet.COM"
```

   This entry will exist in */usr/mmdf/table/root.dom*:

```
seine.mynet.COM:        seine.mynet.COM
```

   These entries will exist in */usr/mmdf/table/local.dom*:

```
seine:          seine.mynet.COM
yangtze:        yangtze.mynet.COM
mekong:         mekong.mynet.COM
```

These entries will exist in */usr/mmdf/table/smtp.chn*:

```
yangtze.mynet.COM       132.147.118.5
mekong.mynet.COM        132.147.118.6
seine.mynet.COM         132.147.118.4
```

As a result of the initial mail configuration, these entries will exist in */usr/mmdf/table/local.chn*:

```
seine:                  seine
seine.UUCP:             seine
seine.mynet.COM         seine
```

Also, the directory */usr/spool/mmdf/lock/home/q.badhosts* is created with owner and group *mmdf*.

5. Repeat steps 1–4 on *yangtze*. When you are finished, these entries will exist in */usr/mmdf/mmdftailor*:

```
MLDOMAIN        "mynet.COM"
MLNAME          "yangtze"

MDMN name="root", dmn="", show="Root Domain", table=rootdom
MDMN name="localdom", dmn="mynet.COM", show="mynet.COM Domain",
     table=localdom

MCHN show="Local delivery", name=local, que=local, tbl=local, pgm=local,
     ap=822, mod=imm
MCHN show="SMTP Delivery", name=smtp, que=smtp, tbl=smtpchn, pgm=smtp,
     ap=822, mod=host, confstr="charset=7bit"
MCHN show="Smart-host Routing for hosts", name=badhosts, que=badhosts,
     tbl=smtpchn, pgm=smtp, ap=822, mod=host,
     confstr="hostname=yangtze.mynet.COM", host="mekong.mynet.COM"
```

The directory */usr/spool/mmdf/lock/home/q.badhosts* exists with owner and group *mmdf*.

These entries will exist in */usr/mmdf/table/local.dom*:

```
yangtze:        yangtze.mynet.COM
seine:          seine.mynet.COM
mekong:         mekong.mynet.COM
```

These entries will exist in */usr/mmdf/table/smtp.chn*

```
seine.mynet.COM         132.147.118.4
mekong.mynet.COM        132.147.118.6
yangtze.mynet.COM       132.147.118.5
```

As a result of the initial mail configuration, these entries will exist in */usr/mmdf/table/local.chn*:

```
yangtze:                yangtze
yangtze.UUCP:           yangtze
yangtze.mynet.COM       yangtze
```

6. Repeat steps 1–3 on *mekong*. When you are finished, these entries will exist in */usr/mmdf/mmdftailor*:

```
MLDOMAIN        "mynet.COM"
MLNAME          "mekong"

MDMN name="root", dmn="", show="Root Domain", table=rootdom
MDMN name="localdom", dmn="mynet.COM", show="mynet.COM Domain",
     table=localdom

MCHN show="Local delivery", name=local, que=local, tbl=local, pgm=local,
     ap=822, mod=imm
MCHN show="SMTP Delivery", name=smtp, que=smtp, tbl=smtpchn, pgm=smtp,
     ap=822, mod=host, confstr="charset=7bit"
```

   The directory */usr/spool/mmdf/lock/home/q.badhosts* exists with owner and group *mmdf*.

   These entries will exist in */usr/mmdf/table/local.dom*:

```
yangtze:        yangtze.mynet.COM
seine:          seine.mynet.COM
mekong:         mekong.mynet.COM
```

   This entry will exist in */usr/mmdf/table/smtp.chn*

```
seine.mynet.COM         132.147.118.4
yangtze.mynet.COM       132.147.118.5
mekong.mynet.COM        132.147.118.6
```

   As a result of the initial mail configuration, these entries will exist in */usr/mmdf/table/local.chn*:

```
mekong:                 mekong
mekong.UUCP:            mekong
mekong.mynet.COM        mekong
```

When you have finished, MMDF is configured on all three machines so that *seine* and *yangtze* will forward all unrecognized local mail to *mekong*, which knows about both machines.

An advantage to this configuration is that if you were to add another machine (for example, *columbia*) to the network, you would only have to configure *columbia* and change the setup on *mekong*. This is simpler than the configuration shown in "Setting up MMDF on a TCP/IP network" (page 142), in which you would have to reconfigure every system on the network. Mail should be addressed using Internet style addresses, such as *root@seine.mynet.COM*.

Setting up MMDF over TCP/IP with a UUCP Internet gateway

This section describes setting up MMDF on a TCP/IP-based network of machines, connecting to the Internet via a UUCP gateway machine. In this case, all machines on the ethernet will forward all mail to one machine, which will then distribute the mail to the appropriate destination.

This section assumes that the network machines are configured as described in "Setting up MMDF using a mail gateway" (page 144).

1. Configure UUCP to add your Internet gateway (for example, *uunet*). See Chapter 7, "Connecting to other computers with UUCP" in the *System Administration Guide*.
2. Because you are assuming the configuration in "Setting up MMDF using a mail gateway" (page 144), the only machine you have to change is *mekong.mynet.COM*. Log in as *mmdf* on *mekong*.
3. Run the **MMDF Configuration Manager**. Select the UUCP network and provide the full domain name for your gateway (for example, *uunet.uu.net*) when prompted.
4. In the **MMDF Configuration Manager**, specify that you want to forward mail addressed to unknown hosts (page 73) over the **badhosts** channel (page 91) to the smart host (the gateway), *uunet.UU.NET*.

   This line will exist in */usr/mmdf/table/root.dom*:

   ```
   uunet.UU.NET:   uunet.UU.NET
   ```

   This line will exist in */usr/mmdf/table/uucp.dom*:

   ```
   uunet:  uunet.UUCP
   ```

   These lines will exist in */usr/mmdf/table/uucp.chn*:

   ```
   uunet.UU.NET:   uunet!%s
   ```

   These lines will exist in */usr/mmdf/mmdftailor*:

   ```
   MLDOMAIN        "mekong.COM"
   MLNAME          "mekong"

   UUname  mekong

   MCHN show="Smart-host Routing for hosts", name=badhosts, que=badhosts,
        tbl=uuchn, pgm=uucp, ap=822, mod=imm,
        confstr="hostname=mekong.mynet.COM", host=uunet.uu.NET
   ```

   The directory */usr/spool/mmdf/lock/home/q.badhosts* exists with owner and group *mmdf*.

When you have finished, MMDF is configured to forward all unrecognized mail originating in the *mynet.COM* domain to *uunet.UU.NET*, the Internet gateway. Mail should be addressed using Internet style addresses, such as *root@seine.mynet.COM*.

See also:

- "Specifying format for mail user addresses" (page 71)

Chapter 5

sendmail administration

The **sendmail**(ADMN) utility is a mail router provided by SCO. The current version of **sendmail**(ADMN) is version 8.8.8, which includes various responses to CERT security alerts. You do not need to use **sendmail** to route mail; the operating system provides a default mail router, MMDF, which provides several key benefits over using **sendmail**. For a comparison to MMDF, see "Comparison of sendmail with MMDF" (page 8).

sendmail

- implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.
- is a mail router that collects messages generated by a Mail User Agent (MUA) such as **mail**(C), determines a route for delivery, edits the message header as required by the destination and delivery agent, and calls an appropriate delivery agent (called a mailer) to deliver the mail. Typical delivery mechanisms include TCP/IP (using SMTP), UUCP, x.400, and local delivery programs such as **lmail**.
- supports either Internet-style addressing, such as user@domain, or UUCP-style addressing, such as host!user.
- supports delivery of mail with X.400-style addressing to an X.400 gateway machine if such a machine is specified in the configuration.
- supports anti-spamming. See "Managing spam" (page 170)
- handles text format files only. Binary data is not allowed unless included in the mail message as a **uuencode**d file. See the **uuencode**(C) manual page.

- is based on:

  RFC 822 (Internet Mail Format Protocol)
  RFC 821 (Simple Mail Transport Protocol)
  RFC 1123 (Internet Host Requirements)
  RFC 1425 (SMTP Service Extensions)

  However, in many cases **sendmail** can be configured to exceed these protocols, as described in this chapter.

- does not provide a user interface for composing and reading mail, nor does it do actual mail delivery. These tasks are handled by the MUA and the mailer, respectively.

- is difficult to configure manually. This implementation of **sendmail**, however, includes a script that creates a basic configuration that is adequate for most sites. For additional information on how to manually fine-tune this configuration, see: "Tuning sendmail configuration" (page 153) and "Advanced sendmail configuration" (page 188)

See also:

- "How sendmail works" (page 179)
- "Standard sendmail configuration" (this page)
- "Reconfiguring sendmail" (page 162)
- "Administering sendmail" (page 162)
- "For more about sendmail" (page 220)

Standard sendmail configuration

Execute the command **mkdev cf** to perform **sendmail** configuration. The major tasks of **mkdev cf** are to:

- lead you through an interactive configuration utility for entering information needed for the **sendmail** configuration file, such as the host and domain name of the local system, the identity (if present) of a UUCP gateway and UUCP connections, and the identity (if present) of an X.400 gateway
- combine the information you enter with a default configuration source file to build and install a standard **sendmail** configuration file. If successful, the configuration file */usr/lib/sendmail.cf* is created.

The default configuration created here supports both UUCP and Internet style addressing. The user entered information allows a network administrator to designate a machine on the network as the UUCP gateway machine, which will process all UUCP requests for the net or subnet. If an X.400 gateway is

specified by the user information, **sendmail** also will forward mail that is addressed in X.400 style to that gateway. The user information also provides for telling **sendmail** to use NIS maps to define mail aliases.

Running mkdev cf

When ready to configure **sendmail**, enter:

mkdev cf.

A menu with nine items appears. **sendmail** remembers the user inputs entered the last time **mkdev cf** was run and displays those entries as the current state/value. The following describes the menu items, including which are mandatory and which are optional.

1. **Edit UUCP Connections**
   If you wish this host to forward mail to other machines via UUCP, you must enter the names of those machines here. You can enter them manually or let this script get them with the **uuname** command.

2. **Edit Domain**
   This is a mandatory item. You must either enter a domain name manually (for example, *sco.com*) or let this script assign it using the **hostname** command.

3. **NIS Support**
   Network Information Service (NIS) is a networked system administration service that provides a facility (maps) for maintaining mail aliases. See "How NIS works" in the *Networking Guide* for more information. If you wish to use NIS maps to define aliases, answer yes to the initial prompt under this menu item. The script will run the **domainname** command to obtain the NIS domain name. If domainname returns successfully, this script displays the NIS domain name. If it is unsuccessful, it will warn you that NIS does not appear to be installed. You may quit this configuration to install NIS or you may complete the remainder of this configuration, then install NIS, and then reconfigure **sendmail**, executing this step again.

4. **Edit Alternate Host Names**
   This menu item is optional; but if mail is addressed to your machine by names **other** than that returned by the **hostname** or **uuname** utilities, you must enter these names here in order to receive that mail.

5. **Miscellaneous Items**
   This menu item displays a submenu of four items regarding network connections to this host. All these network connections have a default status of no. If the status of any of these should be yes, select that item and respond accordingly. When the status of these four items is correct for this host, select menu item 5 to return to the main menu.

6. **Set up X.400 Gateway Configuration**
   This menu item is optional; but if you want to send mail addressed in X.400 format from the local host, you must enter the name of an X.400 gateway machine to which **sendmail** can forward this mail for delivery.

7. **Review configuration information**
   This menu item displays the current state or values of the configuration parameters set with items one to six.

8. **Generate sendmail.cf file**
   This item first invokes menu item #7. After you press ⟨Enter⟩ it prompts you with this message:

   ```
   Do you wish to change anything? [y/n]
   ```

   If you enter **y** (yes), it returns you to the main menu. If you enter **n** (no), it generates the file */usr/lib/sendmail.cf* from one through six menu items and from a default configuration source file. If */usr/lib/sendmail.cf* already exists, this script saves the old *sendmail.cf* file as */usr/lib/sendmail.cf-*. When finished, it displays that the new file has been installed, it kills the **sendmail** daemon if one is running, and starts a new daemon. It then returns you to the main menu.

9. **Quit** Use this item to quit after generating the new configuration file or if you want to interrupt the configuration process.

Editing the daemon invocation

In the standard **sendmail** configuration, one instance of **sendmail** runs as a daemon that typically serves two purposes:

1. It serves as the SMTP server.
2. It forks a sub-daemon at specified intervals to process the mail queue.

This daemon starts when the system enters multi-user mode and stops when the system is shut down. The command that starts this daemon is in the */etc/rc2.d/P86sendmail* script. The default command in this file sets the queue-processing intervals to 1 hour. You do not need to change this command unless you know that you want to change the interval or add other flags.

Running and testing sendmail

To run **sendmail** in its entirety after running **mkdev cf**, start the **sendmail** SMTP server daemon using the command in the */etc/rc2.d/P86sendmail* script. If you do not start this daemon, **sendmail** will only deliver outgoing messages that are not queued and incoming messages from UUCP that are not queued.

Test **sendmail** by creating and sending mail messages with **mail**(C). There are three sources for information about **sendmail** processing:

- error messages which **sendmail** mails to the mail sender if mail is undeliverable
- the logging mechanism; messages are logged through the **syslogd**(ADM) facility (the location of log files is defined in */etc/syslog.conf*) and the logging level is set as an option either in the configuration file or as an argument to **sendmail**
- debugging levels selected as an argument passed to **sendmail**

Tuning sendmail configuration

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file or by editing the **sendmail** daemon invocation in the */etc/rc2.d/P86sendmail* script. For example, the line "OT3d" sets option "T" to the value "3d" (three days). See "O—set option" (page 204) for a complete listing of these options.

Most of these options have defaults appropriate for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for that load. In particular, sites experiencing a large number of small messages, many of which are delivered to many recipients, may find that they need to adjust the parameters dealing with queue priorities.

When making any configuration file changes, see "Reconfiguring sendmail" (page 162).

Timeout options syntax

All options that specify time intervals use a set of predefined units:

s seconds

m minutes

h hours

d days

w weeks

For example, "10m" represents ten minutes, whereas "2h30m" represents two hours and thirty minutes.

Changing the queue processing interval

The **sendmail** daemon started by the */etc/rc2.d/P86sendmail* script sets to one hour the interval at which this daemon will fork a process to process the mail queue. You can change this interval by changing the value assigned to the **-q** argument to the **sendmail** command in the */etc/rc2.d/P86sendmail* script. This interval is typically set to between fifteen minutes and one hour.

Creating a user information database

The user database is built from a text file using the **makemap** utility. The text file is a series of lines corresponding to user database records; each line has a key and a value separated by white space. The key is always in the format described in "Structure of the user database" (this page). for example:

```
eric:maildrop
```

This text file is normally installed in a system directory; for example, it might be called */etc/userdb*. To make the database version of the map, enter:

makemap btree /etc/userdb.db < /etc/userdb

Then reference the database file in the **sendmail** configuration file. For example, include the following line in the configuration file:

OU/etc/userdb

sendmail adds the *.db* suffix when doing the database search.

Structure of the user database

The database is a sorted (BTree-based) structure. User records are stored with the key:

user-name:field-name

The sorted database format ensures that user records are clustered together. Meta-information is always stored with a leading colon.

Field names define both the syntax and semantics of the value. Currently, the defined fields include:

"maildrop" The delivery address for this user. There may be multiple values of this record. In particular, mailing lists will have one **maildrop** record for each user on the list.

"mailname" The outgoing mailname for this user. For each outgoing name, there should be an appropriate **maildrop** record for that name to allow return mail. Also see **:default:mailname** information in "User information database" (page 184).

Altering read timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. These timeouts are set using the **r** option in the configuration file. The argument is a comma-separated list of ***keyword=value*** pairs. The recognized keywords, their default values, and the minimum values allowed by RFC 1123 section 5.3.2 are:

initial	The wait for the initial 220 greeting message [5m, 5m].
helo	The wait for a reply from a HELO or EHLO command [5m, unspecified]. This may require a host name lookup, so five minutes is probably a reasonable minimum.
mail†	The wait for a reply from a MAIL command [10m, 5m].
rcpt†	The wait for a reply from an RCPT command [1h, 5m]. This should be long because it could be pointing at a list that takes a long time to expand.
datainit†	The wait for a reply from a DATA command [5m, 2m].
datablock†	The wait for reading a data block (that is, the body of the message). [1h, 3m]. This should be long because it also applies to programs piping input to **sendmail** which have no guarantee of promptness.
datafinal†	The wait for a reply from the dot terminating a message. [1h, 10m]. If this is shorter than the time actually needed for the receiver to deliver the message, duplicates will be generated. This is discussed in RFC 1047.
rset	The wait for a reply from a RSET command [5m, unspecified].
quit	The wait for a reply from a QUIT command [2m, unspecified].
misc	The wait for a reply from miscellaneous (but short) commands such as NOOP (no-operation) and VERB (go into verbose mode). [2m, unspecified].
command†	In server SMTP, the time to wait for another command. [1h, 5m].

For compatibility with old configuration files, if no "keyword=" is specified, all the timeouts marked with † are set to the indicated value.

Many of the RFC 1123 minimum values may well be too short. **sendmail** was designed to the RFC 822 protocols, which did not specify read timeouts; hence, **sendmail** does not guarantee to reply to messages promptly. In particular, a RCPT command specifying a mailing list will expand and verify the entire list; a large list on a slow system may take more than five minutes. (This verification includes looking up every address with the name server; this involves network delays and can be considerable in some cases.) A one

hour timeout is recommended. Because this failure is rare, a long timeout is not onerous and may ultimately help reduce network load.

For example, the following line sets the server SMTP command timeout to 25 minutes and the input data block timeout to three hours.

```
Orcommand=25m,datablock=3h
```

Altering message timeouts

After sitting in the queue for a few days, a message times out. This means that if a message cannot be delivered for some reason, it is returned to the sender. This ensures that at least the sender is aware that the message was not sent. The timeout is typically set to three days. This timeout is set using the **T** option in the configuration file.

The time reference for this option is the time of submission, which is set in the queue, rather than the amount of time left until timeout. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example:

/usr/lib/sendmail -oT1d -q

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

Because this option is global, and because you do not know how long another host outside your domain will be down, a five day timeout is recommended. This allows a recipient to fix the problem even if it occurs at the beginning of a long weekend. RFC 1123 section 5.3.1.1 says that this parameter should be "at least 4–5 days".

The **T** option can also take a second timeout indicating a time after which a warning message should be sent; the two timeouts are separated by a slash. For example, the following value causes email to fail after five days, but a warning message will be sent after four hours:

5d/4h

This should be large enough that the message will have been tried several times.

Forking during queue runs

When the **Y** option is set, **sendmail** forks before each individual message while running the queue. This prevents **sendmail** from consuming large amounts of memory, and so it may be useful in memory-poor environments. However, if the **Y** option is not set, **sendmail** keeps track of hosts that are down during a queue run, which can improve performance dramatically.

If the **Y** option is set, **sendmail** can not use connection caching.

Altering queue priorities

Every message is assigned a priority when it is first instantiated, consisting of the message size (in bytes) offset by the message class multiplied by the "work class factor" and the number of recipients multiplied by the "work recipient factor." The priority is used to order the queue. Higher numbers for the priority mean that the message is processed later, when running the queue.

The message size is included so that large messages are penalized relative to small messages. The message class allows users to send high-priority messages by including a "Precedence:" field in their message; the value of this field is looked up in the **P** lines of the configuration file. Because the number of recipients affects the size of load a message presents to the system, this is also included in the priority.

The recipient and class factors can be set in the configuration file by using the **y** and **z** options, respectively. They default to 30,000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

```
pri = msgsize - (class * z) + (nrcpt * y)
```

(Remember, higher values for this parameter actually mean that the job is treated with lower priority.)

The priority of a job can also be adjusted each time it is processed (that is, each time an attempt is made to deliver it) using the "work time factor," set by the **Z** option. This is added to the priority, so it normally decreases the precedence of the job, on the grounds that jobs that have failed many times tend to fail again in the future. The **Z** option defaults to 90,000.

Setting the mail load limit

sendmail can be asked to queue (but not deliver) mail if the system load average gets too high using the **x** option. When the load average exceeds the value of the **x** option, the delivery mode is set to **q** (queue only) if the "Queue Factor" (**q** option) divided by the difference in the current load average and the **x** option plus one exceeds the priority of the message—that is, the message is queued if:

pri > q divided by (LA-x+1)

The **q** option defaults to 600,000, so each point of load average is worth 600,000 priority points (as described above).

For drastic cases, the **X** option defines a load average at which **sendmail** will refuse to accept network connections. Locally generated mail (including incoming UUCP mail) is still accepted.

Setting the delivery mode

There are a number of delivery modes for **sendmail**, and they are set by the **d** configuration option. These modes specify how quickly mail is delivered. Legal modes are:

i deliver interactively (synchronously)

b deliver in background (asynchronously)

q queue only (do not deliver)

There are tradeoffs. Mode "i" passes the maximum amount of information to the sender, but it is hardly ever necessary. Mode "q" puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode "b" is probably a good compromise. However, this mode can cause large numbers of processes if you have a mailer that takes a long time to deliver a message.

If you run in mode "q" (queue only), **sendmail** will not expand aliases and follow *.forward* files upon initial receipt of the mail. This speeds up the response to RCPT commands.

Changing file permissions

There are several files involved with **sendmail** that can have a number of modes. The modes depend on the functionality you want and the level of security you require.

suid root options

The **sendmail** program can safely be made setuid to *root*. At the point where it is about to **exec**(S) a mailer, it checks to see if the user ID is zero. If so, it resets the user ID and groupid to a default (set by the **u** and **g** options). (This can be overridden by setting the **S** flag to the mailer for mailers that are trusted and must be called as *root*.)

Temporary file modes

The mode of all temporary files that **sendmail** creates is determined by the **F** option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it is not necessary to run **sendmail** as *root* at all (even when running the queue).

Alias database modes

The recommended *aliases* database mode is 644. While this is not as flexible as if the database were mode 666, it avoids potential security problems with a globally writable database.

The database that **sendmail** actually uses is represented by the file */usr/lib/mail/aliases.db*. The mode on this file should match the mode on */usr/lib/mail/aliases*. If *aliases* is writable and the *aliases.db* file is not, users will be unable to reflect their desired changes through to the actual database. However, if *aliases* is read-only and the *aliases.db* file is writable, a slightly sophisticated user can arrange to steal mail anyway.

If your file is not writable by the world or you do not have auto-rebuild enabled (with the **D** option), then you must be careful to reconstruct the alias database each time you change the text version via the command:

/usr/bin/newaliases

If this step is ignored or forgotten, any intended changes are also ignored or forgotten. See "Maintaining the alias database" (page 168) for more information on the alias database.

Setting connection caching parameters

When processing the queue, **sendmail** will try to keep the last few open connections open to avoid startup and shutdown costs. This only applies to IPC connections.

When trying to open a connection, the cache is first searched. If an open connection is found, it is probed to see if it is still active by sending a NOOP command. It is not an error if this fails; instead, the connection is closed and reopened.

Two parameters control the connection cache. The **k** option defines the number of simultaneous open connections that will be permitted. If it is set to zero, connections will be closed as quickly as possible. The default is one. This should be set as appropriate for your system size; it will limit the amount of system resources that **sendmail** will use during queue runs.

The **K** option specifies the maximum time that any cached connection will be permitted to idle. When the idle time exceeds this value the connection is closed. This number should be small (under ten minutes) to prevent you from grabbing too many resources from other hosts. The default is five minutes.

Using sendmail with a name server

If your system is configured to use the Domain Name Service, then **sendmail** will use it.

However, if you do not have a name server configured at all, such as at a UUCP-only site, **sendmail** will get a "connection refused" message when it tries to connect to the name server (either indirectly by calling **gethostbyname** or directly by looking up MX records). If the **I** option is set, **sendmail** will

interpret this to mean a temporary failure and will queue the mail for later processing; otherwise, it ignores the name server data. If your name server is running properly, the setting of this option is not relevant; however, it is important that it be set properly to make error handling work properly.

This option also allows you to modify name server options. The command line takes a series of flags as documented in **resolver**(SLIB) (with the leading "RES_" deleted). Each can be preceded by an optional "+" or "–". For example, the following line turns on the AAONLY (accept authoritative answers only) and turns off the DNSRCH (search the domain path) options.

```
OITrue +AAONLY -DNSRCH
```

Most resolver libraries default to DNSRCH, DEFNAMES, and RECURSE flags on and all others off. Note the use of the initial "True" - this is for compatibility with previous versions of **sendmail**, but is not otherwise necessary.

Version level 1 configurations turn DNSRCH and DEFNAMES off when doing delivery lookups, but leave them on everywhere else. Version 8 of **sendmail** ignores them when doing canonification lookups (that is, when using $[... $]), and always does the search. If you do not want to do automatic name extension, do not call $[... $].

The search rules for $[... $] are somewhat different than usual. If the name (that is, the "...") has at least one dot, it always tries the unmodified name first. If that fails, it tries the reduced search path, and lastly tries the unmodified name (but only for names without a dot, because names with a dot have already been tried). This allows names such as "utc.CS" to match the site in Czechoslovakia rather than the site in your local computer science department. It also prefers A and CNAME records over MX records - that is, if it finds an MX record it makes note of it, but keeps looking. This way, if you have a wildcard MX record matching your domain, it will not assume that all names match.

Moving the per-user forward files

Some sites mount each user's home directory from a local disk on their workstation, so that local access is fast. However, the result is that *.forward* file lookups are slow. In some cases, mail can even be delivered on machines inappropriately because of a file server being down. The performance can be especially bad if you run the automount facility.

The **J** option allows you to set a path of forward files. For example, the following configuration file line would first look for a file with the same name as the user's login in */usr/forward*; if that is not found (or is inaccessible) the file *.forward* in the user's home directory is searched.

```
OJ/usr/forward/$u:$z/.forward
```

A truly perverse site could also search by sender by using **$r**, **$s**, or **$f**.

If you create a directory such as */usr/forward*, it should be mode 1777 (that is, the sticky bit should be set). Users should create the files mode 644.

Setting mail queue filesystem free space

You can specify a minimum number of free blocks on the queue filesystem using the **b** option. If there are fewer than the indicated number of blocks free on the filesystem on which the queue is mounted, the SMTP server will reject mail with the 452 error code. This invites the SMTP client to try again later.

Beware of setting this option too high; it can cause rejection of email when that mail would be processed without difficulty.

This option can also specify an advertised "maximum message size" for hosts that speak ESMTP.

Setting privacy flags

The **p** option allows you to set certain "privacy" flags. Many of them do not give you any extra privacy, rather just insisting that client SMTP servers use the HELO command before using certain commands.

The option takes a series of flag names; the final privacy is the inclusive "or" of those flags. For example:

```
Opneedmailhelo, noexpn
```

This insists that the HELO or EHLO command be used before a MAIL command is accepted and disables the EXPN command.

The **restrictmailq** option restricts printing the queue to the group that owns the queue directory. It does not make sense to set this if you do not also protect the logs.

The **restrictqrun** option restricts people running the queue (that is, using the **-q** command-line flag) to root and the owner of the queue directory.

Setting "send to me too" operation

Normally, **sendmail** deletes the (envelope) sender from any list expansions. For example, if *matt* sends to a list that contains *matt* as one of the members, he will not get a copy of the message. If the **-m** (me too) flag is passed on the command line, or if the **m** option is set in the configuration file, this behavior is suppressed. Some sites like to run the SMTP daemon with **-m**.

Reconfiguring sendmail

You can change the **sendmail** configuration at any time. How you change the configuration depends on what you want to change. When **sendmail** is installed, a standard configuration file is created by combining user prompted input with non-prompted configuration information. **sendmail** obtains the non-prompted configuration information from a default configuration source file, */usr/lib/sendmail.d/sendmail.src*.

If you want to change any of the information for which **mkdev cf** prompts, the easiest approach is to rerun **mkdev cf** and respond to the prompts. **sendmail** remembers the previous responses to these prompts, so you need only make changes. For a description of the information for which **mkdev cf** prompts, see "Running mkdev cf" (page 151).

If you want to change information for which **mkdev cf** does not prompt:

1. Back up the default configuration source file, */usr/lib/sendmail.d/sendmail.src*, if you want to retain online an original version of this file.
2. Change */usr/lib/sendmail.d/sendmail.src* according to your needs.
3. Run **mkdev cf** to rebuild the */usr/lib/sendmail.cf* file, integrating the changed */usr/lib/sendmail.d/sendmail.src*. See "Running mkdev cf" (page 151) for a description of **mkdev cf**.

CAUTION You should not make changes directly to the */usr/lib/sendmail.cf* file because your changes will be lost if **mkdev cf** is run afterwards.

Administering sendmail

System administration with **sendmail** consists of:

- debugging sendmail (page 163),
- monitoring the system log file (page 163),
- managing the mail queue (page 164),
- maintaining the system alias file (page 168),
- managing spam (page 170),
- and performing other related actions.

Debugging sendmail

There is a fairly large number of debug flags built into **sendmail**. Each debug flag has a number and a level, where higher levels cause more information to be printed out. The convention is that levels greater than nine are not required. They print out so much information that you would not normally want to see them, except for debugging that particular piece of code. Debug flags are set using **sendmail**'s **-d** command-line flag. The syntax 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

The spaces are for reading ease only. For example:

-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.

Viewing the system log

The system log appears in the file logged to by the **syslogd**(ADM) utility. All messages from **sendmail** are logged under the **LOG_MAIL** facility. Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the network), the word "sendmail," and a message.

A large amount of information can be logged. The log is arranged as a succession of levels. At the lowest level, only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under 10 are considered "useful"; log levels above 64 are reserved for debugging purposes. Levels from 11-64 are reserved for verbose information that some sites might want.

The log level is set with the **L** configuration option.

Logging traffic

Many SMTP implementations do not fully implement the protocol. For example, some personal computer based SMTPs do not understand continuation lines in reply codes. These can be very hard to trace. If you suspect such a problem, you can set traffic logging using the **-X** flag. For example, the following will log all traffic in the file */tmp/traffic*.

/usr/lib/sendmail -X /tmp/traffic -bd

This logs a lot of data very quickly and should never be used during normal operations. After starting up such a daemon, force the errant implementation to send a message to your host. All message traffic in and out of **sendmail**, including the incoming SMTP traffic, will be logged in this file.

Dumping state

You can ask **sendmail** to log a dump of the open files and the connection cache by sending it a **SIGUSR1** signal. The results are logged at **LOG_DEBUG** priority.

Viewing the mail queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although **sendmail** ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

The contents of the queue can be printed using the **mailq** command (or by specifying the **-bp** flag to **sendmail**):

mailq

This produces a listing of the queue identifiers, the size of the message, the date the message entered the queue, and the sender and recipients.

Format of sendmail queue files

sendmail queue files live in the directory defined by the **Q** option in the */usr/lib/sendmail.cf* file, usually */usr/spool/mqueue*. All queue files have the form ***x*f*AAA99999***, where *AAA99999* is the ID for this message and the *x* is a type. The first letter of the ID encodes the hour of the day that the message was received by the system (with A being the hour between midnight and 1:00AM). All files with the same ID collectively define one message.

The types are:

- d data file. The message body (excluding the header) is kept in this file.
- l lock file. If this file exists, the job is currently being processed, and a queue run does not process the file. For that reason, an extraneous *lf* file can cause a job to seem to disappear. It will not even time out. This file is obsolete in this release because this system supports **flock** and **lockf** system calls.
- n this file is created when an ID is being created. It is a separate file to ensure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time. (This is only used on old versions of **sendmail**; it is not used on newer versions.)
- q queue control file. This file contains the information necessary to process the job.
- t temporary file. This is an image of the *qf* file when it is being rebuilt. It should be renamed to a *qf* file very quickly.
- x transcript file, existing during the life of a session, showing everything that happens during that session

The *qf* file is structured as a series of lines, each beginning with a code letter. The lines are as follows:

- D name of the data file. There can only be one of these lines.
- H header definition. There can be any number of these lines. The order is important: it represents the order in the final message. This uses the same syntax as header definitions in the configuration file.
- C 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 */etc/passwd* file); ***aliasname*** is the name of the alias that expanded to this address (used for printing messages).
- R recipient address. This is normally completely aliased, but is actually re-aliased when the job is processed. There is one line for each recipient.
- S sender address. There can only be one of these lines.
- E error address. If any such lines exist, they represent the addresses that should receive error messages.
- T job creation time. This computes how long a job remains in the queue undelivered, before being returned to the sender.
- P current message priority. This orders the queue. Higher numbers mean lower priorities. The priority changes as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M message. This line is printed by the **mailq** command, and is generally used to store status information. It can contain any text.

F flag bits, represented as one letter per flag. Defined flag bits are **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.

$ a macro definition. The values of certain macros (as of this writing, only **$r** and **$s**) are passed through to the queue run phase.

B the body type. The remainder of the line is a text string defining the body type. If this field is missing, the body type is assumed to be "undefined" and no special processing is attempted. Legal values are **7BIT** and **8BITMIME**.

As an example, the following is a queue file sent to *"eric@mammoth.Berkeley.EDU"* and *"bostic@okeeffe.CS.Berkeley.EDU"*. (This example is contrived and probably inaccurate for your environment. Glance over it to get an idea; nothing can replace looking at what your own system generates.)

```
P835771
T404261372
DdfAAA13557
Seric
Eowner-sendmail@vango.CS.Berkeley.EDU
Ceric:sendmail@vango.CS.Berkeley.EDU
Reric@mammoth.Berkeley.EDU
Rbostic@okeeffe.CS.Berkeley.EDU
H?P?return-path: <owner-sendmail@vango.CS.Berkeley.EDU>
Hreceived: by vango.CS.Berkeley.EDU (5.108/2.7) id AAA06703;
        Fri, 17 Jul 92 00:28:55 -0700
Hreceived: from mail.CS.Berkeley.EDU by vango.CS.Berkeley.EDU (5.108/2.7)
        id AAA06698; Fri, 17 Jul 92 00:28:54 -0700
Hreceived: from [128.32.31.21] by mail.CS.Berkeley.EDU (5.96/2.5)
        id AA22777; Fri, 17 Jul 92 03:29:14 -0400
Hreceived: by foo.bar.baz.de (5.57/Ultrix3.0-C)
        id AA22757; Fri, 17 Jul 92 09:31:25 GMT
H?F?from: eric@foo.bar.baz.de (Eric Allman)
H?x?full-name: Eric Allman
Hmessage-id: <9207170931.AA22757@foo.bar.baz.de>
HTo: sendmail@vango.CS.Berkeley.EDU
Hsubject: this is an example message
```

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

Queue interval

The amount of time between forking a process to run through the queue is defined by the **-q** flag. If you run **sendmail** in delivery mode **i** or **b**, this can be relatively large, because it is only relevant when a host that was down comes back up. If you run **sendmail** in delivery mode **q**, it should be relatively short, because it defines the amount of time that a message may sit in the queue before **sendmail** tries to deliver it. The delivery mode is a configuration option.

RFC 1123 section 5.3.1.1 says that this value should be at least 30 minutes (although that probably does not make sense if you use "queue-only" mode).

Forcing the queue

The **sendmail** program should run the queue automatically at intervals. The algorithm is to read and sort the queue, then to attempt to process all jobs in order. When it attempts to run the job, **sendmail** first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to ensure that only one queue processor exists at any time, because there is no guarantee that a job cannot take forever to process (however, **sendmail** does include heuristics to try to abort jobs that are taking absurd amounts of time; technically, this violates RFC 821, but is blessed by RFC 1123). Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no completely general way to resolve this.

In some cases, you may find that if a major host goes down for a couple of days, this can create a prohibitively large queue. This situation causes **sendmail** to spend an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later, when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
cd /usr/spool
mv mqueue omqueue; mkdir mqueue; chmod 700 mqueue
```

You should then kill the existing daemon (because it is still processing in the old queue directory) and create a new daemon. To kill the existing daemon and create a new daemon, reboot the system.

To run the old mail queue, run the following command:

```
/usr/lib/sendmail -oQ/usr/spool/omqueue -q
```

The **-oQ** flag specifies an alternate queue directory, and the **-q** flag says just to run every job in the queue. Use the **-v** flag to view what is going on.

When the queue is finally emptied, you can remove the directory:

rmdir /usr/spool/omqueue

You can also limit the jobs to those with a particular queue identifier, sender, or recipient using one of the queue modifiers. For example, "**-qRberkeley**" restricts the queue run to jobs that have the string "berkeley" somewhere in one of the recipient addresses. Similarly, **-qS*sender*** limits the run to a particular sender and **-q*Iidentifier*** limits it to a particular queue identifier.

Maintaining the alias database

The alias database exists in two forms. One is a text form, maintained in the file */usr/lib/mail/aliases*. The aliases are of the form:

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

Only local names can be aliased. For example:

```
eric@prep.ai.MIT.EDU: eric@CS.Berkeley.EDU
```

This does not have the desired effect. Aliases can be continued by starting any continuation line with a space or a tab. Blank lines and lines beginning with a number sign ("#") are comments.

The second form is processed by the **dbm**(S) or db library. This form is in the files */usr/lib/mail/aliases.db*. This is the form that **sendmail** actually uses to resolve aliases. This technique improves performance.

You can also use NIS-based alias files. For example, the following specification will first search the */usr/lib/aliases* file and then the map named *mail.aliases* in *my.nis.domain*.

```
OA/usr/lib/aliases
OAnis:mail.aliases@my.nis.domain
```

WARNING If you build your own NIS-based alias files, be sure to provide the **-l** flag to **makedbm**(NADM) to map uppercase letters in the keys to lowercase; otherwise, aliases with uppercase letters in their names will not match incoming addresses.

Additional flags can be added after the colon exactly like a **K** line. For example:

```
OAnis:-N mail.aliases@my.nis.domain
```

This will search the appropriate NIS map and always include null bytes in the key.

Rebuilding the alias database

The DBM version of the database can be rebuilt explicitly by executing the command:

/usr/bin/newaliases

This is equivalent to giving **sendmail** the **-bi** flag:

/usr/lib/sendmail -bi

If the "**D**" option is specified in the configuration, **sendmail** rebuilds the alias database automatically, if possible, when it is out of date. Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes start the rebuild process simultaneously.

If you have multiple aliases databases specified, the **-bi** flag rebuilds all the database types it understands (for example, it can rebuild db databases but not NIS databases).

Potential alias database problems

There are a number of problems that can occur with the alias database. They all result when a **sendmail** process accesses the DBM version while it is only partially built. This can happen under two circumstances: either one process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

The **sendmail** program includes two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process and leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form:

```
@: @
```

(Note that this is not normally legal.) Before **sendmail** accesses the database, it checks to ensure that this entry exists. (The **a** option is required in the configuration for this action to occur. This should normally be specified.)

For an additional potential problem involving file modes see "Alias database modes" (page 158).

List owners

If an error occurs on sending to a certain address, say "*x*," **sendmail** looks for an alias of the form "owner-*x*" to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
mail-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
        sam@matisse
owner-mail-wizards: eric@ucbarpa
```

This would cause *eric@ucbarpa* to get the error that occurs when someone sends to *mail-wizards*, due to the inclusion of *nosuchuser* on the list.

List owners also cause the envelope sender address to be modified. The contents of the owner alias are used if they point to a single user, otherwise the name of the alias itself is used. For this reason, and to obey Internet conventions, a typical scheme would be:

```
list:   some, set, of, addresses
list-request:   list-admin-1, list-admin-2, ...
owner-list:     list-request
```

Managing spam

Unsolicited email, or "spam", is an increasing problem on the Internet. You can use anti-spam rulesets implemented in **sendmail** version 8.8.8 to:

- prevent your site from being used as an intermediate site between a sender and a recipient, using check_rcpt (page 171)
- prevent mail from being sent from a pre-defined list of fully qualified domain names and/or IP numbers, regardless of recipient, using check_relay (page 173)
- require that the domain of the sender specified in the "Mail From:" SMTP command resolves to a valid fully-qualified domain name, using check_mail (page 174)
- prevent mail from being sent from a pre-defined list of domain names or email addresses to a specified list of recipients, using check_compat (page 174)

These anti-spam features of **sendmail** are disabled by default. You must manually modify the **sendmail** configuration file, */usr/lib/sendmail.cf*, file to configure and enable these rulesets.

Configuring anti-spam features requires a thorough knowledge of both SMTP (RFC 821) and **sendmail**. The O'Reilly book on **sendmail**, listed in "Related documentation" (page 2), provides detailed information on **sendmail** configuration. Other resources are also available at *http://www.sendmail.org*.

NOTE The ruleset names are built into **sendmail**, and do not always indicate the functionality they enable. See the descriptions of each ruleset for a detailed description of their capabilities.

Currently, anti-spam features cannot be enabled with the SCOadmin **Sendmail Configuration Manager**, **mkdev cf**, or the **Internet Manager**'s mail configuration options. Modifications made to */usr/lib/sendmail.cf* to enable anti-spam features are not preserved across successive executions of these configuration utilities. Be sure to save your changes to */usr/lib/sendmail.cf* so that you can re-enable the anti-spam features should you run one of the **sendmail** configuration utilities.

Using the check_rcpt ruleset

Those sending spam mail often try to use an intermediate system in an attempt to hide the source of electronic mail. The check_rcpt ruleset prevents your site from being used as an intermediate site between a sender and a recipient.

To implement check_rcpt:

1. Uncomment the check_rcpt, Parse0, and matchvdom rulesets, as well as the Krelays line, in */usr/lib/sendmail.cf*.
2. Add entries to the *relays* file (page 172) for those sites that are allowed to send mail through this site.
3. Restart **sendmail** (page 176).

After your restart **sendmail**, attempts to relay mail through this site are rejected and an error message is returned to the sender, unless:

- the site is contained in the relays map entries in */usr/lib/mail/antispam/relays*, or
- the recipient address is in the same domain as this system
- the recipient is in the domain of one of the virtual domains served by this system

If no extra sites will use this site as a relay, do not add entries to the relays map.

NOTE This ruleset will account for and allow mail delivery to virtual domains defined by the SCO **Internet Manager**, if you uncomment the appropriate section of */usr/lib/sendmail.cf*.

The class R, defined by entries in the file */usr/lib/mail/antispam/sendmail.cR*, allows additional relays not defined in the relays map. You must also have DD and Cw defined for this ruleset to function properly.

Adding entries to the relays map

The relays file (*/usr/lib/mail/antispam/relays*), used by the the check_rcpt ruleset, specifies those sites and IP numbers that are allowed to use this site as an intermediate relay.

Add entries to the file using this ⟨Tab⟩-separated format:

address `OK`

address is either the fully-qualified domain name or the IP number of the site that is allowed to use this one as an intermediate relay. The fields must be ⟨Tab⟩-separated, and the `OK` entry is required.

For example, to allow the site *bomb20.pdev.sco.com* to use this site as an intermediate relay, add the following line to the file:

```
bomb20.pdev.sco.com     OK
```

This example shows a specific IP number:

```
132.147.67.15   OK
```

After adding or deleting entries from this file, rebuild the relays map:

1. Log in to the system as *root*.
2. Enter the following commands:

   cd /usr/lib/mail/antispam
   makemap hash relays < relays

Using the check_relay ruleset

This ruleset prevents mail from being sent from a pre-defined list of fully qualified domain names and/or IP numbers, regardless of recipient.

To implement this feature:

1. Uncomment the check_relay ruleset and the Kspammers line in */usr/lib/sendmail.cf*.
2. Modify and rebuild the spammers file (this page).
3. Restart **sendmail** (page 176).

When implemented, mail sent from the sites indicated in */usr/lib/mail/antispam/spammers* will be returned to the sender with an error message.

NOTE Do not use both check_relay and check_mail. The check_mail ruleset is essentially a superset of check_relay. However, check_mail may be too strict when dealing with remote SMTP clients that may not be configured properly in the Domain Name Service.

Adding entries to the spammers map

The spammers file (*/usr/lib/mail/antispam/spammers*), used by the check_relay or check_mail rulesets, identifies systems from which mail will be rejected.

Add entries to the file using this ⟨Tab⟩-separated format:

address message

address is either the fully qualified domain name or the IP number of the site from which this system will refuse mail. ***message*** is the error message to be sent back to the sender. For example, to refuse mail from the machines *bomb20.pdev.sco.com* and the machine at IP address *132.147.67.15*, you might add these entries:

```
bomb20.pdev.sco.com     Mail rejected, contact postmaster@mydomain.com
132.147.67.15           Mail rejected, contact postmaster@mydomain.com
```

Note that the error message is only used by check_mail, not check_relay. However, a string must always exist on the right hand side of this file regardless of which ruleset uses it.

After editing this file, rebuild the spammers map:

1. Log in to the system as *root*.
2. Enter the following commands:

   cd /usr/lib/mail/antispam
   makemap hash spammers < spammers

Using the check_mail ruleset

This ruleset requires that the domain of the sender specified in the "Mail From:" SMTP command resolves to a valid fully-qualified DNS domain name. Additionally, the client making the connection to the local SMTP server is checked against a pre-defined list of fully qualified domain names.

To implement this feature:

1. Uncomment the check_mail ruleset and the Kspammers line from */usr/lib/sendmail.cf*.
2. Modify and rebuild the spammers file (page 173).
3. Restart **sendmail** (page 176).

After you implement this feature, SMTP "Mail From:" commands whose argument is not a valid fully qualified domain name (as listed by the Domain Name Service), will cause that SMTP transaction to terminate with an error. Additionally, if the "Mail From:" check succeeds, the IP number and name of the client making the connection will be checked against the domains and IP numbers listed in */usr/lib/mail/antispam/spammers*, and the mail returned with an error message if a match is found.

NOTE Do not use both check_relay and and check_mail. The check_mail ruleset is essentially a superset of check_relay, However, check_mail may be too strict when dealing with remote SMTP clients that may not be configured properly in the Domain Name Service. Also, check_mail is incompatible with the **DeliveryMode=Defer** option, as it requires an immediate DNS lookup to verify. (**DeliveryMode=Defer** is not set by default).

Using the check_compat ruleset

Use check_compat to prevent mail from being sent from a pre-defined list of domain names or email addresses to a specified list of recipients. For example, you may use this ruleset for preventing all mail from any user in domain *foobar.com* from being sent to any user in domain *barfoo.com*, but still allow mail from users in *foobar.com* to be sent to users in other domains. This is useful for combating individual spam attacks from individual sites to a specific set of users or domains.

To implement this feature:

1. Uncomment the check_compat, matchprotected, and matchspam rulesets, and the Kspammers2 and Kprotected lines, from */usr/lib/sendmail.cf*.
2. Add users and domains from which mail will possibly be rejected to the *spammers2* file and rebuild the map (page 175).

3. Add users and domains that are considered 'protected' from spam attacks from the first set of addresses to the *protected* file and rebuild the map (page 176).
4. Restart **sendmail** (page 176).

Adding entries to the spammers2 map

The spammers2 file (*/usr/lib/mail/antispam/spammers2*), used by the check_compat ruleset, specifies those addresses that are to be considered as potential spammers against those addresses in the protected map.

Add entries to the file using this (Tab)-separated format:

```
address SPAMMER
```

address is the user name, system name, or domain name which is considered a source of spam mail. For example, if mail from all users in *isendspam.com* are to be considered generators of spam mail, enter:

```
isendspam.com          SPAMMER
```

This will mark as a potential spam attack all mail from all users in the domain *isendspam.com*, as well as all of its subdomains such as *machine1.isendspam.com* and *machine1.subdom.isendspam.com* are all considered possible spam.

To specify an individual user instead, enter their individual addresses:

```
chris@sendyouspam.com  SPAMMER
```

This marks *chris@sendyouspam.com* as a potential spam generator, but does not affect mail from other users in *sendyouspam.com*.

All entries must contain the string `SPAMMER` on the right hand side.

After editing this file, rebuild the spammers2 map:

1. Log in to the system as *root*.
2. Enter the following commands:

   cd /usr/lib/mail/antispam
   makemap hash spammers2 < spammers2

NOTE Use both the *protected* and *spammers2* files carefully.

Because you can block whole domains from access to your protected users, you may also exclude valid e-mail addresses. In this case, it is best to target individual addresses in the *spammers2* file.

Adding entries to the protected users map

The protected users file (*/usr/lib/mail/antispam/protected*), used by the check_compat ruleset, specifies those addresses that are to be considered 'protected' from spam attacks by the addresses in the *spammers2* map.

Add entries to the file using this ⟨Tab⟩-separated format:

address `PROTECTED`

address is the user name, system name, IP address, or domain name which is considered protected. For example, if mail to all users in *foobar.com* are protected, enter the line:

```
foobar.com      PROTECTED
```

This will mark as protected all mail to all users in the domain *foobar.com*, as well as its subdomains such as machine1.foobar.com, and machine1.subdom.foobar.com.

To protect individual users rather than entire domains, enter their individual addresses:

```
chris                       PROTECTED
chris@foobar.com            PROTECTED
```

This marks as protected the local user *chris* and the address *chris@foobar.com*, but leaves as unprotected all other local users and all other users in the domain *foobar.com*.

All entries must contain the string `PROTECTED` on the right hand side.

After editing this file, rebuild the protected map:

1. Log in to the system as *root*.
2. Enter the following commands:

   cd /usr/lib/mail/antispam
   makemap hash protected < protected

Stopping and restarting sendmail

To stop and restart **sendmail**:

1. Log in to the system as *root*.
2. Enter the following commands:

   cd /etc/rc2.d
   ./P86sendmail stop
   ./P86sendmail start

Enabling user forwarding (.forward files)

As an alternative to the alias database, any user can put a file with the name *.forward* in his or her home directory. If this file exists, **sendmail** redirects mail for that user to the list of addresses listed in the *.forward* file. For example, suppose the home directory for user *mckee* has a *.forward* file with contents:

```
mckee@ernie
kirk@calder
```

Then any mail arriving for *mckee* is redirected to the specified accounts.

Actually, the configuration file defines a sequence of filenames to check. By default, this is the user's *.forward* file, but can be defined to be more files by using the **J** option. If you change this, you will have to inform your user base of the change; *.forward* is the standard place to define mail redirection.

Special header lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into **sendmail** that cannot be changed. These built-ins are described here.

Errors-to:

If errors occur anywhere during processing, this header causes error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

The "Errors-To:" header was created when UUCP did not understand the distinction between an envelope and a header; this was a work-around that provided what should now be passed as the envelope sender address. It should go away. It is only used if the **l** option is set.

Apparently-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 any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC 822.

Summary of support files

This is a summary of the support files that **sendmail** creates or generates. Many of these can be changed by editing the *sendmail.cf* file; these are noted below.

/usr/lib/sendmail	binary of **sendmail**; should be setuid root.
/usr/bin/newaliases	link to */usr/lib/sendmail*; causes the alias database to be rebuilt. Running this program is completely equivalent to giving **sendmail** the **-bi** flag.
/usr/bin/mailq	prints a listing of the mail queue. This program is equivalent to using the **-bp** flag to **sendmail**.
/usr/lib/sendmail.cf	configuration file, in textual form.
/usr/lib/sendmail.hf	help file used by the SMTP HELP command; the actual path of this file is defined in the **H** option of the *sendmail.cf* file.

/usr/lib/sendmail.st	statistics file. If you wish to collect statistics about your mail traffic, you should create this file:
	`cp /dev/null /usr/lib/sendmail.st`
	`chmod 666 /usr/lib/sendmail.st`
	Print this file with the command **mailstats**(ADMN). The actual path of this file is defined in the **S** option of the *sendmail.cf* file.

/etc/sendmail.pid	process id file. Created in daemon mode; contains the process id of the current SMTP daemon. If you use this in scripts, use "head -1" to get just the first line; later versions of **sendmail** may add information to subsequent lines.
/usr/lib/mail/aliases	textual version of the alias file.
/usr/lib/mail/aliases.db	alias file in db format.
/usr/spool/mqueue	directory in which the mail queue and temporary files reside; this directory should be mode 700 and owned by root; the actual path of this directory is defined in the **Q** option of the *sendmail.cf* file.
*/usr/spool/mqueue/qf**	control (queue) files for messages.
*/usr/spool/mqueue/df**	data files.
*/usr/spool/mqueue/tf**	temporary versions of the *qf* files, used during queue-file rebuild.
*/usr/spool/mqueue/xf**	transcript of the current session.

How sendmail works

This section provides an overview of how mail is processed and delivered on an SCO system when **sendmail** is chosen as the mail router (in place of MMDF) and **sendmail** is used in the standard configuration. In the standard configuration, **sendmail** runs as a daemon (acting as the SMTP server listening for SMTP requests) and as one or more user processes (or "instances") that process and deliver mail messages.

Outgoing mail

Being a mail router, **sendmail** does not provide a user interface for composing and sending mail. These tasks are handled by a Mail User Agent (MUA). The standard MUA on SCO systems is the **mail**(C) program. Others include **scomail**(XC) and **SCO Shell Mail** (page 39).

With **sendmail** configured:

1. The user composes a piece of mail with an MUA such as **mail**(C).
2. **execmail** transfers the mail to **sendmail** by executing a new instance of **sendmail**.

   execmail passes any command-line flags and options it received from **mail** on to **sendmail**. These flags and options, along with the contents of the */usr/lib/sendmail.cf* file, control the operations of this instance of **sendmail**.
3. **sendmail** collects the recipient address(es) prior to collecting the entire message and parses the address(es) to determine the appropriate mailer and any aliases, forwarding addresses, or expanded address lists.
4. **sendmail** collects the message.
5. **sendmail** either stores the message in the queue for delivery later or calls upon the appropriate mailer to deliver the message now.

If additional MUAs are added to the system, they may request mail routing from **sendmail** by executing, using **exec**(S), a new instance of **sendmail** directly without going through **execmail**.

Incoming mail

sendmail processes and delivers incoming mail the same way it processes and delivers outgoing mail (this page); the only difference being the source of the mail. **sendmail** receives incoming mail from other hosts via servers. The standard **sendmail** configuration supports two servers: SMTP and UUCP.

SMTP server

One instance of **sendmail**, generally invoked by the */etc/rc2.d/P86sendmail* script at boot time, runs as the SMTP server by default. This server listens on the SMTP (Simple Mail Transfer Protocol) port for SMTP requests. When it receives a request, the **sendmail** daemon executes the **fork** system call to start a new instance of **sendmail** to process the request.

UUCP server

A standard SCO distribution includes a UUCP mail server called **rmail**(ADMN). This standard program routes incoming mail from UUCP to **sendmail** by executing a new instance of **sendmail**.

sendmail execution

Each time **sendmail** starts up, it reads a configuration file that specifies the operations of that instance of **sendmail**. A **sendmail** configuration file includes:

- definitions of mailers **sendmail** can use to route mail
- rules for parsing addresses
- definitions for rewriting message headers
- settings of various options

Options may also be set on the command line by passing them as arguments to **sendmail**. An option set on the command line overrides any setting of that option in the configuration file. For a complete list of options see "O—set option" (page 204).

Command-line flags, different from the options and passed to **sendmail** as arguments, also influence processing. See the **sendmail**(ADMN) manual page for a description of these flags.

Using options and flags

sendmail options control operations such as whether **sendmail** delivers mail immediately after processing or queues the mail for delivery when the queue is next processed. Options set in the configuration file can also be overridden by passing options to **sendmail** as arguments.

Flags determine things such as the interval at which **sendmail** processes the mail queue and whether to use an alternate configuration file for this instance of **sendmail**.

These options and flags alter **sendmail**'s processes of address parsing, message collection, and storage. Part of the address parsing process is the rewriting of addresses. Some additional rewriting may also take place during **sendmail**'s delivery phase.

Configuration file priority

sendmail looks in two places for its configuration file and uses the highest priority file available. The highest priority is a filename passed as a flag argument to the **sendmail** daemon. If a filename is passed to **sendmail** using the **-C** flag, **sendmail** reads this file for configuration information and ignores all others.

If the **-C** flag is not specified on the command line, **sendmail** reads */usr/lib/sendmail.cf*. This is the standard configuration file created by default when initializing **sendmail**.

Mailers

sendmail hands off mail by executing a mailer with arguments as defined in the *mailer* definition in the configuration file.

The mailers in the standard **sendmail** configuration are:

tcp used to route mail via the Internet by calling the SMTP server on the recipient machine of the mail recipient.

uucp used to route mail via UUCP by executing **uux**(C).

local used to deliver mail to recipients on the local host by executing **lmail**(ADMN). **lmail** delivers the mail to the recipient's mailbox file called ***recipient_name*** at the location */usr/spool/mail/recipient_name*. The recipient extracts the mail from this file using an MUA such as **mail**.

prog used to deliver mail to a program (using an invocation of the shell). This is explained further in "Mail to files and programs" (page 188).

x400 used to deliver X.400-style addressed mail from the local system to the X.400 gateway machine specified in the configuration file. In the standard **sendmail** configuration, this mailer calls the program **/usr/lib/sendxmail** on the X.400 gateway machine.

If a mailer returns a status that indicates it cannot handle the message now but might be able to handle it later, **sendmail** puts the message in the mail queue and tries again to deliver it the next time the mail queue is processed. If a message is not deliverable (which can be caused by unknown recipients, a delivery timeout, a destination mailer that refuses to accept the message, or errors during processing), **sendmail** notifies the sender and returns the message. The notification includes an error code when possible.

Recipient address parsing

Recipient addresses are passed to **sendmail** as arguments or via the SMTP **RCPT** command depending on the interface to **sendmail**. Each address is parsed in order to build a recipient list for the message being processed. Parsing involves resolving each address to an internally used form that identifies the mailer to be used. If the mailer identified for any address is the "local" mailer, the following additional processing occurs:

- **sendmail** checks to see if the address matches an entry in the alias database; if so, it expands the alias and appends the new address to the recipient list.
- If no alias is found (or if the alias points back to the same address), **sendmail** appends ":maildrop" to the addressee name to form a lookup key and then searches the user information database (page 184). If a match in the database is found, **sendmail** appends the new address to the recipient list.
- If no match in the user information database occurs (or if the maildrop points to the same address), **sendmail** checks to see if the user has a **$HOME**/*.forward* file. If so, **sendmail** appends the new addresses found there to the recipient list.

When new recipient addresses are appended to the recipient list through special "local" mailer address processing, the old name is retained in the list and a flag is set that tells the delivery phase to ignore the old name. In this way the recipient list is kept free from duplicates, preventing alias loops and duplicate messages from being delivered to the same recipient, as might occur if a person is in two groups.

At this stage, before the message is collected, the address syntax has been checked and local addresses verified; detailed checking of host names and host addresses, however, is deferred until message delivery.

Address format

Before **sendmail** can deliver a message, it must interpret the address. **sendmail** expects addresses to conform, or be convertible, to a format defined in RFC 822. **sendmail** passes each recipient address through a set of filters called "rewriting rules" to determine whether it understands the address and can deliver to that address via one of its known mailers.

NOTE **sendmail** also recognizes mail addresses in X.400 format, but will not try to interpret or deliver mail to them. Instead, **sendmail** will forward such mail to an X.400 mail gateway, if such a gateway is defined in the configuration file.

The **mailaddr**(ADMN) manual page and the following text provide brief overviews of the RFC 822 format.

1. Anything in parentheses is thrown away (as a comment).
2. Anything in angle brackets (" <> ") is preferred over anything else. This rule implements the Internet standard that addresses of this form will send to the electronic ***machine-address*** rather than the human ***user name***:

   user name *<**machine-address**>*

3. Double quotes (") quote phrases; backslashes quote characters. Backslashes are more powerful because they 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 quotes must be properly balanced and nested. The rewriting rules control remaining parsing.

Special "local" mailer address processing

sendmail applies the following special processing to mail intended for the "local" mailer:

- aliasing (this page)
- user information database lookups (page 184)
- forwarding (page 184)
- inclusion (page 185)

Aliasing and user information database lookups apply across the entire system. Forwarding allows all users to redirect incoming mail destined for their accounts. Inclusion directs **sendmail** to read a file for a list of addresses. Inclusion is normally used in conjunction with aliasing.

Aliasing

Aliasing expands recipient names into address lists using a system-wide text file that associates a recipient name with a list of addresses. Only names that parse as local are allowed as aliases. This guarantees a unique key. The identity of the alias text file is configured through the **sendmail** configuration file and is configured to be */usr/lib/mail/aliases* by the standard configuration. **sendmail** indexes this file to speed access. For more information on the alias files see the following:

- "Alias database modes" (page 158)
- "Maintaining the alias database" (page 168)

User information database

If you have specified one or more databases using the **U** option, the databases will be searched for a "*user*:maildrop" entry. If found, the mail will be sent to the specified address.

If the first token passed to the user part of the "local" mailer is an at sign, the at sign will be stripped off and the database lookup will be skipped. The intent is that the user database will act as a set of defaults for a cluster (for example, the computer science department of a university); mail sent to a specific machine should ignore these defaults.

When mail is sent, the name of the sender is looked up in the database. If that sender has a **mailname** record, the value of that record is used as their outgoing name. For example, *eric* might have a record:

```
eric:mailname    Eric.Allman@CS.Berkeley.EDU
```

This would cause *eric*'s outgoing mail to be sent as *Eric.Allman*.

If a "maildrop" is found for the sender, but no corresponding "mailname" record exists, the record ":default:mailname" is consulted. If present, this is the name of a host to override the local host (for example, *CS.Berkeley.EDU*). The effect is that anyone known in the database gets their outgoing mail stamped as *user@CS.Berkeley.EDU*, but people not listed in the database use the local hostname.

For information on creating the user information database see "Creating a user information database" (page 154).

Forwarding

After aliasing and a user information database lookup, local and valid recipients are checked for the existence of a *.forward* file in their home directory. If one exists, the message is not sent to that user, but rather to the list of users in that file. Often, this list contains a single address and is used only for network mail forwarding. For example, suppose user *dbaker* has a *.forward* file in **$HOME** with contents:

```
dbaker@scribe.npr.com
```

Then any mail arriving for *dbaker* would be directed to *dbaker*'s account on *scribe.npr.com*.

Forwarding also permits a user to specify a private incoming mailer. For example, this forwarding line defines a different incoming mailer:

```
"|/usr/local/newmail myname"
```

Including

The syntax for including a file is:

```
:include: pathname
```

When **sendmail** sees an address in this form, it reads the file specified by *pathname* and sends to all users listed in that file.

The intention is not to support direct use of this feature, but rather to use this as a subset of aliasing. In the following example, use of the "include" address format allows a project with the mail alias *projectC* to maintain a project mailing list without interaction with the system administration, even if the alias file is protected.

```
projectC: :include:/usr/project/userlist
```

It is not necessary to rebuild the index on the alias database when a list of this type is changed. All that is needed is to edit the include file to reflect the changes. In this example, the include file is */usr/project/userlist*.

Message collection and storage

Once all recipient addresses are parsed and verified, **sendmail** collects the message. To simplify the program interface, the message is collected even if no addresses are valid; in this case the message is then returned to the sender with an error.

The message format must conform to RFC 822, which means it must consist of two parts: a message header and a message body, separated by a blank line. Further, RFC 822 requires that the header be in a format shown below. The header is parsed and stored in memory, and the body of the message is saved in a temporary file.

Message header

To conform to RFC 822, the header must be a series of lines of the form:

field-name: field-value

A *field-value* can be split across lines by starting the lines that follow with a space or a tab. Some header fields have special internal meaning, in which case **sendmail** may perform special processing on them. Other headers are simply passed through. Some header fields, such as time stamps, may be added automatically 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. Additional editing of a header may take place during the delivery phase to customize the header to meet mailer requirements.

Message body

No RFC 822 formatting requirements are imposed on the message body, except that they must be lines of text; binary data is not allowed. It is completely uninterpreted and untouched by **sendmail**, 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. (SMTP uses lines beginning with a dot to signal the end of the message.) The message body is stored in a separate file from the header information.

Message delivery

sendmail can be configured to deliver a message at the time it receives the message (immediate delivery) or to queue the message for delivery when the mail queue is processed. Immediate delivery can be configured to happen synchronously or asynchronously (in the background). This section on message delivery applies to all of the above.

The result of the recipient address parsing is a triple consisting of "mailer, host, user" for each recipient. The mailer is one of the defined mailers in the configuration file; host is the destination host; and user is the recipient on that host.

For each unique "mailer,host" pairing in the recipient list, **sendmail** calls the appropriate mailer. **sendmail** will try to batch deliver the message to all recipients on the same host in one invocation of the receiving mailer. Mailers that only accept one recipient at a time are handled accordingly. After a connection with the mailer is established, **sendmail** customizes the message header as necessary for correct interpretation by the recipient mailer and sends the result to the mailer.

The receiving mailer status code is caught and checked. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes. An exit code from the receiving mailer must conform to a system standard; otherwise, **sendmail** forwards a generic message such as `Service unavailable` to the originator of the mail.

If the mailer returns a status code indicating that the message should be sent later, **sendmail** puts the mail on the queue and tries again later.

Return to sender

If errors occur during processing, **sendmail** returns the message to the sender for retransmission. If the user agent (**mail**) detects the error, then it is put in the *dead.letter* file located in the sender's home directory. If a **sendmail** server is connecting with a **sendmail** client on another machine, then the user is presumed to have become detached from the transaction, and so the message is mailed back to them.

Queued messages

If the mailer returns a `temporary failure` exit status, the message is queued. A control file for the message describes the recipients to be sent to 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 significant 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 message file that was originally collected. See "Viewing the mail queue" (page 164) for more information on the mail queue.

sendmail interfaces

There are three ways **sendmail** communicates, both in receiving and in sending mail. These are:

- using the conventional UNIX system argument vector/return status
- speaking SMTP over a pair of UNIX system pipes
- speaking SMTP over a socket

For simplicity, the following descriptions of these three methods assume **sendmail** is sending to a mailer.

Argument vector/exit status

This technique is the standard UNIX system method for communicating with a process. A list of recipients is sent in the argument vector, and the message is sent on the standard input. This is the method by which **sendmail** communicates with **execmail**, **rmail**, **lmail**, and **uux**. Anything that the recipient mailer prints is simply collected and sent back to the mail sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

SMTP over pipes

The SMTP protocol can be used to run an interactive lock-step interface with a mailer over a pair of named pipes. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the mailer's standard input. Anything appearing on the mailer's standard output must be an SMTP reply code. This method is not used in the standard configuration. This method is useful when the mailer is

	a program, such as **lmail**, but the number of recipients is larger than the allowable number of arguments that can be passed in the argument vector.
SMTP over a socket	This method is similar to SMTP over pipes, except that it uses a socket. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a **sendmail** process on another machine. This is the method by which **sendmail** routes and receives mail across the Internet via TCP/IP.

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 maintain a public repository of systems messages (such as the Berkeley **msgs** program).

Any address passing through the initial parsing algorithm as a local address (not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar (" | ") the rest of the address is processed as a shell command. If the user name begins with a slash mark (" / ") the name is used as a file name, instead of a login name.

Files that have setuid or setgid bits set but no execute bits set have those bits honored if **sendmail** is running as root.

Advanced sendmail configuration

Most **sendmail** configuration is accomplished by editing the configuration file, */usr/lib/sendmail.cf*. The syntax of this file is designed to be reasonably easy to parse, because this is done every time **sendmail** starts up. As a result, the configuration file is not particularly easy for a human to read or write.

The configuration file has three major purposes:

- setting up the environment for **sendmail**
- rewriting addresses in the message
- mapping addresses into the set of instructions that will deliver the message

For further details see "Purpose of the configuration file" (page 215).

This section provides an overview of the standard configuration file built with the **mkdev cf** script, provides the specifications for configuration file lines (page 191), and describes building a configuration file from scratch (page 215).

/usr/lib/sendmail.cf overview

A **sendmail** 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 number sign ("#") are comments.

The following paragraphs in this description of the standard configuration file refer to "sections" of the standard configuration file. When viewing the configuration file */usr/lib/sendmail.cf*, these sections are differentiated by comments constructed to look like headers by encircling them with number signs ("#").

Macros

The first two sections of the standard configuration file consist of macro definitions. These are lines that begin with the letter "D". The defined macros are described in "D—define macro" (page 196).

Macros can be used in three ways. They can transmit unstructured textual information into the mail system. An example of this is the name that **sendmail** uses to identify itself in error messages; the default, "MAILER-DAEMON", is defined like this:

```
DnMAILER-DAEMON
```

Macros can transmit information from **sendmail** to the configuration file in creating other fields (such as argument vectors to mailers). Examples are the sender name and the recipient host and user names. Other macros are unused internally. These macros can be used as shorthand in the configuration file.

Other example macros:

```
DlFrom $g $d
Do.:%@!^/[]
```

The first line defines the format of the mail "From" line. This definition results in "From <sender_address> <current_date>". The second line defines the delimiter characters used in addresses (this defines the delimiters ".:%@!^/[]").

Options

The next section in the file consists of options. These are lines that begin with the letter "O". There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, and so forth. See "O—set option" (page 204) for a complete list of these options. Because options can also be set as an argument in the

invocation of **sendmail**, some of the most useful are also listed in the **sendmail**(ADMN) manual page.

Example options:

```
OA/usr/lib/mail/aliases
OT3d
```

The first line defines the location of the alias file (*/usr/lib/mail/aliases* in this option setting). The second line defines the timeout interval for undelivered mail (this option definition specifies that undelivered mail is returned after 3 days).

Message precedences

This next section in the file assigns values to mail precedence names. These values are used to compute a priority for a mail message. **sendmail** uses this priority to order mail messages in the mail queue. See "Altering queue priorities" (page 157) for a description of how precedence values are used.

Example precedence setting:

```
Pspecial-delivery=100
```

This setting defines a value of 100 for messages that include "special-delivery" in the "Precedence:" field. (The lower the value, the higher priority given the message.)

Format of headers

This next section in the file contains header declarations that inform **sendmail** of the format of known header lines. Knowledge of a few header lines is built into **sendmail**, such as the "From:" and "Date:" lines. See "H—define header" (page 203) for a description of how headers are declared and "D—define macro" (page 196). for a description of the macros used.

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.

Example header format:

```
H?D?Date: $a
H?F?From: $q
```

The first line declares a "Date:" header and defines its contents to be the origination date in RFC 822 format. The second line declares a "From:" header and defines its contents to be the sender's address.

Address rewriting rules

The next several sections of the file are the rewriting rules which **sendmail** uses to parse addresses. These are lines beginning with the letter "S", which names the ruleset to which the following rules belong, and the letter "R", which are the rules.

There are multiple rule sets with specific purposes. These purposes are discussed in "Semantics of rewriting rule sets" (page 195).

Mailer declarations

At the end of the standard configuration file are sections that contain the mailer declarations. These are lines beginning with the letter "M". You will see rewriting rule lines between and after the mailer declaration lines. These are for rulesets that customize addresses for specific mailers.

Mailer declarations tell **sendmail** of the various mailers available to it. A mailer declaration specifies the internal name of the mailer, the pathname of the program to call, some of the flags associated with the mailer, and an argument vector to be used in the call to the mailer. See "M—define mailer" (page 200).

Example mailer declaration:

```
Mlocal, P=/usr/bin/lmail, F=lsFDMPuhCE, S=10, R=20, A=lmail $u
```

Configuration file lines

The following sections define the syntax for each key letter used in a **sendmail** configuration file:

- "V—configuration version level" (page 211)
- "K—key file declaration" (page 212)

R and S—rewriting rules

The core of address parsing is the rewriting rules. These are an ordered list of pattern-replacement rules which are applied to each address. The **sendmail** command 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 address 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.

The syntax of these two commands is:

S*n*	Sets the current ruleset being collected to *n*. If you begin a ruleset more than once, it deletes the old definition.
R*lhs 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. The fields must be separated by at least one tab character; there may be embedded spaces in the fields.

Macro expansions of the form **$***x* are performed when the configuration file is read. Expansions of the form **$&***x* are performed at run time using a somewhat less general algorithm. This is intended only for referencing internally defined macros such as **$h** that are changed at runtime.

The left-hand side (LHS)

The left-hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

$*	match zero or more tokens
$+	match one or more tokens
$-	match exactly one token
$=*x*	match any phrase in class *x*
$~*x*	match any word not in class *x*

If any of these match, they are assigned to the symbol **$** ***n*** for replacement on the right-hand side (RHS), where ***n*** is the index in the LHS. For example, suppose a LHS rule of the following:

```
$-:$+
```

Further, suppose the following input:

```
UCBARPA:eric
```

In this example, the input matches the rule, and the values passed to the RHS are:

$1 UCBARPA
$2 eric

Additionally, the LHS can include **$@** to match zero tokens. This is *not* bound to a **$** ***N*** on the RHS and is normally only used when it stands alone in order to match the null input.

The right-hand side (RHS)

When the left-hand side of a rewriting rule matches, the input is deleted and replaced by the right-hand side. Tokens are copied directly from the RHS, unless they begin with a dollar sign. Metasymbols are:

$*n*	substitutes indefinite token ***n*** from LHS
$[*name*$]	canonicalizes ***name***
$(*map key* $@*arguments* $:*default* $)	generalized keyed mapping function
$>*n*	calls ruleset ***n***
$#*mailer*	resolves to ***mailer***
$@*host*	specifies ***host***
$:*user*	specifies ***user***

The **$*n*** syntax substitutes the corresponding value from a **$+**, **$-**, **$***, **$=**, or **$~** match on the LHS. It can be used anywhere.

A host name enclosed between **$[** and **$]** is looked up using the **gethostbyname**(SLIB) routines and replaced by the canonical name. (This is actually completely equivalent to:

$(host *hostname* $)

In particular, a **$:** default can be used.) For example, $[[128.32.130.2]$] might become *vango.CS.Berkeley.EDU*, and $[csam$] might become *lbl-csam.arpa*. **sendmail** recognizes its numeric IP address without calling the name server and replaces it with its canonical name.

The **$(** ... **$)** syntax is a more general form of lookup; it uses a **named** map instead of an implicit map. If no lookup is found, the indicated default is inserted; if no default is specified and no lookup matches, the value is left unchanged.

The **$>** ***n*** syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset ***n***. The final value of ruleset ***n*** then becomes the substitution for this rule.

The **$#** syntax should only be used in ruleset zero or a subroutine of ruleset zero. It causes evaluation of the ruleset to terminate immediately, and it signals to **sendmail** that the address has completely resolved. The complete syntax is:

$#*mailer* **$@*host*** **$:*user***

This specifies the {mailer, host, user} triple necessary to direct the mailer. If the mailer is local, the host part can be omitted. (You may want to use it for special "per user" extensions. For example, at CMU you can send email to "jgm+foo"; the part after the plus sign is not part of the user name, and is passed to the local mailer for local use.) The ***mailer*** must be a single word, but the ***host*** and the ***user*** can be multi-part. If the ***mailer*** is the built-in IPC mailer, the ***host*** may be a colon-separated list of hosts that are searched in order for the first working address (exactly like MX records). The ***user*** is later rewritten by the mailer-specific envelope rewriting set and assigned to the **$u** macro. As a special case, if the value to "$#" is "local" and the first character of the "$:" value is "@", the "@" is stripped off and a flag is set in the address descriptor that causes **sendmail** to not do ruleset 5 processing.

Normally, a rule that matches is retried, that is, the rule loops until it fails.

A RHS can also be preceded by a **$@** or a **$:** to change this behavior. A **$@** prefix causes the ruleset to return with the remainder of the RHS as the value. A **$:** prefix causes the rule to terminate immediately, but the ruleset to continue. This can avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes can precede a **$>** specification. For example, the form:

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

matches anything, passes that to ruleset seven, and continues; the **$:** is necessary to avoid an infinite loop.

Substitution occurs in the order described; that is, parameters from the LHS are substituted, hostnames are canonicalized, "subroutines" are called and, finally, **$#**, **$@**, and **$:** are processed.

Semantics of rewriting rule sets

There are five rewriting sets that have specific semantics.

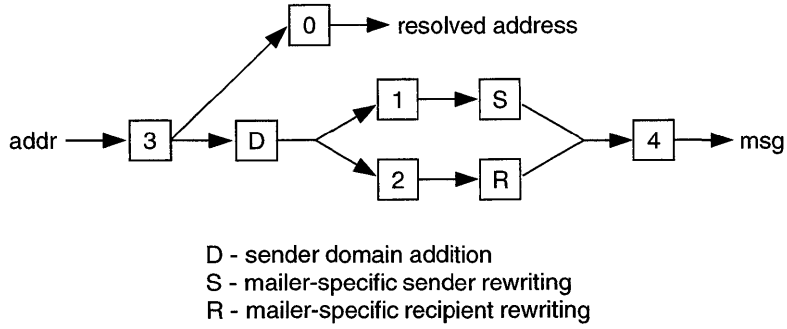


Figure 5-1 Rewriting set semantics

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

```
local-part@host-domain-spec
```

If no "@" sign is specified, then the host-domain-spec can be appended from the sender address (if the **C** flag is set in the mailer definition corresponding to the sending mailer). Ruleset three is applied by **sendmail** before doing anything with any address.

Ruleset zero is applied after ruleset three to addresses 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 argv expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses, respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

IPC mailers

Some special processing occurs if the ruleset zero resolves to an IPC mailer (that is, a mailer that has "[IPC]" listed as the Path in the **M** configuration line. The host name passed after "$@" has MX expansion performed; this looks the name up in DNS to find alternate delivery sites.

The host name can also be provided as a dotted quad in square brackets; for example:

```
[128.32.149.78]
```

This causes direct conversion of the numeric value to a TCP/IP host address.

The host name passed in after the "$@" may also be a colon-separated list of hosts. Each is separately MX expanded and the results are concatenated to make (essentially) one long MX list. The intent here is to create "fake" MX records that are not published in DNS for private internal networks.

As a final special case, the host name can be passed in as a text string in square brackets:

```
[ucbvax.berkeley.edu]
```

This form avoids the MX mapping.

NOTE This is intended only for situations where you have a network firewall, so that your MX record points to a gateway machine; this machine could then do direct delivery to machines within your local domain. Use of this feature directly violates RFC 1123 section 5.3.5; it should not be used lightly.

D—define macro

Macros are named with a single character. These can be selected from the entire ASCII set, but user-defined macros should be selected from the set of uppercase letters only. Lowercase letters and special symbols are used internally.

The syntax for macro definitions is:

D*x val*

Here *x* is the name of the macro and *val* is the value it should have. Macros are interpolated using the construct **$***x*, where *x* is the name of the macro to be interpolated. This interpolation is done when the configuration file is read, except in **M** lines. The special construct **$&***x* can be used in **R** lines to get deferred interpolation.

Conditionals can be specified using the syntax:

$?x if

$| else

$. endif

For example:

$?x *text1* **$|** *text2* **$.**

This interpolates *text1* if the macro **$x** is set, and *text2* otherwise. The "else" (**$|**) clause can be omitted.

Lowercase macro names are reserved to have special semantics, used to pass information in or out of **sendmail**; some special characters are reserved to provide conditionals; and so on. Uppercase names (that is, **$A** through **$Z**) are specifically reserved for configuration file authors.

The following macros are defined and/or used internally by **sendmail** for interpolation into argv's for mailers or for other contexts. The ones marked † are information passed into **sendmail**, the ones marked ‡ are information passed both in and out of **sendmail**, and the unmarked macros are passed out of **sendmail** but are not otherwise used internally.

NOTE As of **sendmail** version 8.6, all of the macros marked † have reasonable defaults. Previous versions required that they be defined.

$a the origination date in RFC 822 format.

$b the current date in RFC 822 format.

$c the hop count.

$d the current date in UNIX system (ctime) format.

$e†** the SMTP entry message. This is printed out when SMTP starts up. The first word must be the **$j macro as specified by RFC 821. Defaults to "**$j Sendmail $v ready at $b**". Commonly redefined to include the configuration version number, for example, "**$j Sendmail $v/$Z ready at $b**".

$f the sender (from) address.

$g the sender address relative to the recipient.

$h the recipient host.

$i the queue id.

$j‡ the "official" domain name for this site. This is fully qualified if the full qualification can be found. It *must* be redefined to be the fully qualified domain name if your system is not configured so that **sendmail** can find it automatically.

$k the UUCP node name (from the **uname** system call).

$l† the format of the mail "From" line. Unless you have changed the UNIX system mailbox format, you should not change the default, which is "**From $g $d**".

$m the domain part of the **gethostname** return value. Under normal circumstances, **$j** is equivalent to "**$w.$m**".

$n† the name of the daemon (for error messages). Defaults to "**MAILER-DAEMON**".

$o†** the set of "operators" in addresses. This macro defines a list of characters which, during parsing, are considered tokens and are considered separators of tokens. For example, if "@" were in the **$o macro, then the input **a@b** would be scanned as three tokens: "a", "@", and "b". Defaults to the characters ".:@[]", which is the minimum set necessary to do RFC 822 parsing; a richer set of operators is ".:%@!/[]", which adds support for UUCP, the %-hack, and X.400 addresses.

$p **sendmail**'s process id.

$q†** default format of sender address. The **$q macro specifies how an address should appear in a message when it is defaulted. Defaults to "<$g>". It is commonly redefined to be

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

or

```
$g$?x ($x)$.
```

which correspond to the following two formats:

Eric Allman <eric@CS.Berkeley.EDU>
eric@CS.Berkeley.EDU (Eric Allman)

sendmail properly quotes names that have special characters if the first form is used.

$r protocol used to receive the message.

$s sender's host name.

$t a numeric representation of the current time.

$u the recipient user.

$v the version number of **sendmail**.

$w‡** the hostname of this site. The **$w macro is set to the root name of this host (but see below for caveats).

$x the full name of the sender.

$z the home directory of the recipient.

$_ the validated sender address.

There are three types of dates that can be used. The **$a** and **$b** macros are in RFC 822 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, **$a** is set to the current time also. The **$d** macro is equivalent to the **$b** macro in UNIX system (ctime) format.

The macros **$w**, **$j**, and **$m** are set to the identity of this host. **sendmail** tries to find the fully qualified name of the host if at all possible; it does this by calling **gethostname**(SLIB) to get the current hostname and then passing that to **gethostbyname**(SLIB) which is supposed to return the canonical version of that host name. (For example, on some systems **gethostname** might return "foo" which would be mapped to *foo.bar.com* by **gethostbyname**.) Assuming this is successful, **$j** is set to the fully qualified name and **$m** is set to the domain part of the name (everything after the first dot). The **$w** macro is set to the first word (everything before the first dot) if you have a level 5 or higher configuration file; otherwise, it is set to the same value as **$j**. If the canonification is not successful, it is imperative that the configuration file set **$j** to the fully qualified domain name. (Earlier versions of **sendmail** did not pre-define **$j** at all, so up until **sendmail** version 8.6, configuration files **always** had to define **$j**.)

The **$f** macro is the ID of the sender as originally determined; when you are mailing to a specific host, the **$g** macro is set to the address of the sender relative to the recipient. For example, if *eric* sends to *bollard@matisse.CS.Berkeley.EDU* from the machine *vango.CS.Berkeley.EDU*, the **$f** macro is "**eric**" and the **$g** macro is "**eric@vango.CS.Berkeley.EDU**".

The **$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as a flag to **sendmail**. The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the */etc/passwd* file.

When sending, the **$h**, **$u**, and **$z** macros are set to the host, user, and home directories (if local) of the recipient. The first two are set from the **$@** and **$:** parts of the rewriting rules, respectively.

The **$p** and **$t** macros create unique strings (for example, for the "Message-Id:" field). The **$i** macro is set to the queue ID on this host; if put into the timestamp line, it can be extremely useful for tracking messages. The **$v** macro is set to be the version number of **sendmail**; this is normally put in timestamps and has proved extremely useful for debugging.

The **$c** macro is set to the "hop count," that is, the number of times this message has been processed. This can be determined by the **-h** flag on the command line or by counting the timestamps in the message.

The **$r** and **$s** macros are set to the protocol used to communicate with **sendmail** and the sending ***hostname***.

The **$_** macro is set to a validated sender host name.

C and F—define classes

Classes of phrases may be defined to match on the left-hand side of rewriting rules, where a "phrase" is a sequence of characters that do not contain space characters. For example, a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes may be given names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

The syntax is:

C*c phrase1 phrase2...*
F*c file*

The first form defines the class *c* to match any of the named words. It is permissible to split them among multiple lines, for example:

```
CHmonet ucbmonet
```

This example is equivalent to the following:

```
CHmonet
CHucbmonet
```

The second form reads the elements of the class *c* from the named *file*.

The **$~** (match entries not in class) only matches a single word; multi-word entries in the class are ignored in this context.

The class **$=w** is set to be the set of all names this host is known by. This can be used to match local hostnames.

The class **$=k** is set to be the same as **$k**, that is, the UUCP node name.

The class **$=m** is set to the set of domains by which this host is known, initially just **$m**.

M—define mailer

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

M *name*,
{*field=value* }*

Here *name* is the name of the mailer (used internally only) and the "*field=name*" pairs define attributes of the mailer. Fields are:

"Path"	pathname of the mailer
"Flags"	special flags for this mailer
"Sender"	rewriting set for sender addresses
"Recipient"	rewriting set for recipient addresses

"Argv"	argument vector to pass to this mailer
"Eol"	end-of-line string for this mailer
"Maxsize"	the maximum message length for this mailer
"Linelimit"	the maximum line length in the message body
"Directory"	the working directory for the mailer

Only the first character of the field name is checked.

The following flags may be set in the mailer description. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers.

a run Extended SMTP (ESMTP) protocol (defined in RFCs 1425, 1426, and 1427).

b force a blank line on the end of a message. This is intended to work around some versions of */bin/mail* that require a blank line, but do not provide it themselves. It would not normally be used on network mail.

c do not include comments in addresses. This should only be used if you have to work around a remote mailer that gets confused by comments.

C If mail is received from a mailer with this flag set, any addresses in the header that do not have an at sign ("@") after being rewritten by ruleset three have the @domain clause from the sender tacked on. This allows mail with headers of the following form to be rewritten automatically; for example:

```
From: usera@hosta
To: userb@hostb, userc
```

becomes:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

D This mailer wants a "Date:" header line.

e This mailer is expensive to connect to, so try to avoid connecting normally. Any necessary connection occurs during a queue run.

E escape lines beginning with "From" in the message with a ">" sign.

f The mailer wants a **-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).

F This mailer wants a "From:" header line.

g Normally, **sendmail** sends internally generated email (for example, error messages) using the null return address as required by RFC 1123. (Actually, this only applies to SMTP, which uses the "MAIL FROM:<>"

command.) However, some mailers do not accept a null return address. If necessary, you can set the **g** flag to prevent **sendmail** from obeying the standards; error messages will be sent as from the MAILER-DAEMON (actually, the value of the **$n** macro).

h Uppercase should be preserved in host names for this mailer.

I This mailer is speaking SMTP to another **sendmail**. As such, it can use special protocol features. This option is not required (that is, if this option is omitted, the transmission still operates successfully, although perhaps not as efficiently as possible).

l This mailer is local (that is, final delivery is performed).

L Limit the line lengths as specified in RFC 821. This deprecated option should be replaced by the "L=" mail declaration. For historic reasons, the **L** flag also sets the **7** flag.

m This mailer can send 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.

M This mailer wants a "Message-Id:" header line.

n Do not insert a UNIX system style "From:" line on the front of the message.

p Use the route-addr style reverse-path in the SMTP "MAIL FROM:" command, rather than just the return address. Although this is required in RFC 821 section 3.1, many hosts do not process reverse-paths properly. Reverse-paths are officially discouraged by RFC 1123.

P This mailer requires a "Return-Path:" line.

r This is the same as **f**, but sends a **-r** flag.

s Strip quote characters off the address before calling the mailer.

S Do not reset the user ID before calling the mailer. This would be used in a secure environment where **sendmail** ran as *root*. This could be used to avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (for example, a user's *mail.cf* file).

u Uppercase should be preserved in user names for this mailer.

U This mailer wants UNIX system style "From" lines with the UUCP-style "remote from <host>" on the end.

x This mailer wants a "Full-Name:" header line.

X This mailer wants to use the hidden dot algorithm as specified in RFC 821. Basically, any line beginning with a dot has an extra dot prepended (to be stripped at the other end). This ensures that lines in the message containing a dot do not terminate the message prematurely.

7 strip all output to seven bits. This is the default if the **L** flag is set. Note that clearing this option is not sufficient to get full eight bit data passed through **sendmail**. If the **7** option is set, this is essentially always set, because the eighth bit was stripped on input.

The mailer with the special name "error" can be used to generate a user error. The (optional) host field is an exit status to be returned, and the user field is a message to be printed. The exit status may be numeric or one of the values USAGE, NOUSER, NOHOST, UNAVAILABLE, SOFTWARE, TEMPFAIL, PROTOCOL, or CONFIG to return the corresponding EX_ exit code. For example, the entry:

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

on the RHS of a rule will cause the specified error to be generated and the "Host unknown" exit status to be returned if the LHS matches. This mailer is only functional in ruleset zero.

The mailer named "local" **must** be defined in every configuration file. This is used to deliver local mail, and is treated specially in several ways. Additionally, three other mailers named "prog", "*file*", and "*include*" may be defined to tune the delivery of messages to programs, files, and ":include:" lists respectively. They default to:

```
Mprog, P=/bin/sh, F=lsD, A=sh -c $u
M*file*, P=/dev/null, F=lsDFMPEu, A=FILE
M*include*, P=/dev/null, F=su, A=INCLUDE
```

The Sender and Recipient rewriting sets may either be a simple integer or may be two integers separated by a slash; if so, the first rewriting set is applied to envelope addresses and the second is applied to headers.

The Directory is actually a colon-separated path of directories to try. For example, the definition "D=$z:/" first tries to execute in the recipient's home directory; if that is not available, it tries to execute in the root of the filesystem. This is intended to be used only on the "prog" mailer, because some shells (such as **csh**) refuse to execute if they cannot read the home directory. Because the queue directory is not normally readable by normal users **csh** scripts as recipients can fail.

H—define header

The format of the header lines that **sendmail** inserts into the message is defined by the **H** line. The syntax of this line is:

H[*?mflags?*]
hname*: *htemplate

Continuation lines in this specification are reflected directly into the outgoing message. The ***htemplate*** is macro-expanded before insertion into the message. If the ***mflags*** (surrounded by question marks) are specified, at least one

of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input, it is reflected to the output, regardless of these flags.

Some headers have special semantics, described in the following sections.

O—set option

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

O *o value*

This sets option *o* to be ***value***. Depending on the option, ***value*** can be a string, an integer, a Boolean (with legal values "t", "T", "f", or "F"; the default is TRUE), or a time interval.

a*N* If set, wait up to ***N*** minutes for an @:@ entry to exist in the alias database before starting up. If it does not appear in ***N*** minutes, rebuild the database (if the **D** option is also set) or issue a warning.

A*spec, spec, ...*
Specify possible alias file(s). Each ***spec*** should be in the format "***class***: ***file***" where "***class***:" is optional and defaults to "implicit". Several valid classes are "implicit" and built-in to **sendmail**. These are: "hash", "stab", (internal symbol table—not normally used unless you have no other database lookup), and "nis".

If a list of ***spec***s are provided, **sendmail** searches them in order.

b{***N*** | ***M***} Insist on at least ***N*** blocks free on the filesystem that holds the queue files before accepting email via SMTP. If there is insufficient space, **sendmail** gives a 452 response to the MAIL command. This invites the sender to try again later. The optional ***M*** is a maximum message size advertised in the ESMTP EHLO response. It is currently otherwise unused.

B*c* Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to space (that is, no change is made).

c If an outgoing mailer is marked as being expensive, do not connect immediately.

C*N* Checkpoints the queue every ***N*** (default 10) addresses sent. If your system crashes during delivery to a large list, this prevents retransmission to any but the last ***N*** recipients.

d*x* Deliver in mode ***x***. Legal modes are:

- i Deliver interactively (synchronously).
- b Deliver in background (asynchronously).

q Just queue the message (deliver during queue run).

Defaults to "b" if no option is specified, "i" if it is specified but given no argument (that is, "Od" is equivalent to "Odi").

D If set, rebuild the alias database if necessary and possible. If this option is not set, **sendmail** will never rebuild the alias database unless explicitly requested using **-bi**.

e*x* Dispose of errors using mode *x*. The values for *x* are:

p Print error messages (default).

q No messages, just give exit status.

m Mail back errors.

w Write back errors (mail if user not logged in).

e Mail back errors and give zero exit status always.

E{*/file* | *message*}

Prepend error messages with the indicated message. If it begins with a slash, it is assumed to be the pathname of a file containing a message (this is the recommended setting). Otherwise, it is a literal message. The error file might contain the name, email address, and/or phone number of a local postmaster who could provide assistance to end users. If the option is missing or null, or if it names a file which does not exist or which is not readable, no message is printed.

This option can only be used successfully by user *root*.

f Save UNIX system style "From" lines at the front of headers. Normally, they are assumed redundant and discarded.

F*mode* The file mode for queue files.

g*n* Set the default group ID in which mailers are to run to *n*. Defaults to 1. The value can also be given as a symbolic group name.

G Allow fuzzy matching on the "GECOS" field. If this flag is set, and the usual user name lookups fail (that is, there is no alias with this name and a **getpwnam** fails), sequentially search the password file for a matching entry in the "GECOS" field. This also requires that "MATCHGECOS" be turned on during compilation. This option is not recommended.

h*N* The maximum hop count. Messages that have been processed more than *N* times are assumed to be in a loop and are rejected. Defaults to 25.

H*file* Specify the help file for SMTP.

i Ignore dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

I Insist that the BIND name server (see the **named**(ADMN) manual page) be running to resolve host names. If this is not set and the name server is not running, the */etc/hosts* file will be considered complete. In general, you do want to set this option if your */etc/hosts* file does not include all hosts known to you or if you are using the MX (mail forwarding) feature of the BIND name server. The name server will still be consulted even if this option is not set, but **sendmail** will feel free to resort to reading */etc/hosts* if the name server is not available. Thus, you should *never* set this option if you do not run the name server.

j If set, send error messages in MIME format (see RFC 1341 and RFC 1344 for details).

J*path* Set the path for searching for users' *.forward* files. The default is "$z/.forward". Some sites that use the automount facility may prefer to change this to "/usr/forward/$u" to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons; **sendmail** stops at the first file it can successfully and safely open. For example, "/usr/forward/$u:$z/.forward" will search first in */usr/forward/**username*** and then in *~**username**/.forward* (but only if the first file does not exist).

k*N* The maximum number of open connections that will be cached at a time. The default is one. This delays closing the current connection until either this invocation of **sendmail** needs to connect to another host or it terminates. Setting it to zero defaults to the old behavior, that is, connections are closed immediately.

K*timeout* The maximum amount of time a cached connection will be permitted to idle without activity. If this time is exceeded, the connection is immediately closed. This value should be small (on the order of ten minutes). Before **sendmail** uses a cached connection, it always sends a NOOP (no operation) command to check the connection; if this fails, it reopens the connection. This keeps your end from failing if the other end times out. The point of this option is to be a good network neighbor and avoid using up excessive resources on the other end. The default is five minutes.

l If there is an "Errors-To:" header, send error messages to the addresses listed there. They normally go to the envelope sender. Use of this option causes **sendmail** to violate RFC 1123.

L*n* Set the default log level to ***n***. Defaults to 9.

m Send to me, too, even if I am in an alias expansion.

M*x value* Set the macro *x* to *value*. This is intended only for use from the command line.

n Validate the RHS of aliases when rebuilding the alias database.

o Assume that the headers may be in old format; that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parentheses, 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.

O*options* Set server SMTP options. The options are *key=value* pairs. Known keys are:

Port	Name/number of listening port (defaults to "smtp")
Addr	Address mask (defaults INADDR_ANY)
Family	Address family (defaults to INET)
Listen	Size of listen queue (defaults to 10)

The address mask (that is, **Addr** value) may be a numeric address in dot notation or a network name.

p *opt,opt,...*

Set the privacy *opt*ions. "Privacy" is really a misnomer; many of these are just a way of insisting on stricter adherence to the SMTP protocol. The values of *opt* can be selected from:

public	Allow open access
needmailhelo	Insist on HELO or EHLO command before MAIL
needexpnhelo	Insist on HELO or EHLO command before EXPN
noexpn	Disallow EXPN entirely
needvrfyhelo	Insist on HELO or EHLO command before VRFY
novrfy	Disallow VRFY entirely
restrictmailq	Restrict **mailq** command
restrictqrun	Restrict **-q** command-line flag
goaway	Disallow essentially all SMTP status queries
authwarnings	Put X-Authentication-Warning: headers in messages

The **goaway** pseudo-flag sets all flags except **restrictmailq** and **restrictqrun**. If the **mailq** command is restricted (that is, **restrictmailq** is specified), only people in the same group as the queue directory can print the queue. If queue runs are restricted,

only root and the owner of the queue directory can run the queue. **authwarnings** adds warnings about various conditions that may indicate attempts to spoof the mail system, such as using a non-standard queue directory.

P*postmaster*
: If set, copies of error messages will be sent to the named ***postmaster***. Only the header of the failed message is sent. Because most errors are user problems, this is probably not a good idea on large sites and arguably contains all sorts of privacy violations.

q*factor*
: Use ***factor*** as the multiplier in the map function to decide when just to queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (**x** flag) to determine the maximum message priority that is sent. Defaults to 60,000.

Q*dir*
: Use the specified ***dir*** as the queue directory.

r*timeouts*
: Timeout reads after ***time*** interval. The ***timeouts*** argument is a list of ***keyword=value*** pairs. The recognized timeouts, their default values, and their minimum values specified in RFC 1123 section 5.3.2 are:

initial	wait for initial greeting message [5m, 5m]
helo	reply to HELO or EHLO command [5m, none]
mail	reply to MAIL command [10m, 5m]
rcpt	reply to RCPT command [1h, 5m]
datainit	reply to DATA command [5m, 2m]
datablock	data block read [1h, 3m]
datafinal	reply to final "." in data [1h, 10m]
rset	reply to RSET command [5m, none]
quit	reply to QUIT command [2m, none]
misc	reply to NOOP and VERB commands [2m, none]
command	command read [1L, 5m]

All but **command** apply to client SMTP. For backward compatibility, a timeout with no "keyword=" part will set all of the longer values.

R
: Normally, **sendmail** tries to eliminate any unnecessary explicit routes when sending an error message (as discussed in RFC 1123, section 5.2.6). For example, when sending an error message to

<@known1,@known2,@unknown:user@known3>

sendmail will strip off the *@known1* in order to make the route as direct as possible. However, if the **R** option is set, this will be

disabled, and the mail will be sent to the first address in the route, even if later addresses are known. This may be useful if you are caught behind a firewall.

s Be extra safe when running things; that is, always instantiate the queue file, even if you are going to attempt immediate delivery. The **sendmail** program always instantiates the queue file before returning control to the client.

S*file* Log statistics in the specified *file*.

t*tzinfo* Set the local time zone info to *tzinfo*—for example, "PST8PDT". Actually, if this is not set, the **TZ** environment variable is cleared (so the system default is used); if set but null, the user's **TZ** variable is used, and if set and non-null the **TZ** variable is set to this value.

The current implementation of **sendmail** obtains the identity of the local time zone from the operating system. The time zone is displayed in a mail message header as a field on the "Date" line. It appears as a numeric offset from Coordinated Universal Time (UCT, formerly known as GMT). As described in RFC 822, the time zone field consists of an offset character (+ or -) followed by a four-digit offset. The first two digits represent hours and the last two represent minutes. As an example, the time zone -0600 represents 6 hours earlier than UCT. A time zone of +0430 represents 4 ½ hours ahead of UCT.

T{ *rtime* | *wtime* }
Set the queue timeout to *rtime*. After this interval, messages that have not been successfully sent are returned to the sender. Defaults to five days. The optional *wtime* is the time after which a warning message is sent. If it is missing or zero then no warning messages are sent.

u*n* Set the default user ID for mailers to *n*. Any mailer without the *S* flag in the mailer definition runs as this user. Defaults to 1. The value can also be given as a symbolic user name.

U*udbspec* The user database specification.

v Run in verbose mode. If this is set, **sendmail** adjusts options **c** (do not connect to expensive mailers) and **d** (delivery mode) so that all mail is delivered completely in a single job so that you can see the entire delivery process. Option **v** should *never* be set in the configuration file; it is intended for command line use only.

V*fallbackhost*
If specified, the *fallbackhost* acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity.

w — If you are the "best" (that is, lowest preference) MX for a given host, you should normally detect this situation and treat that condition specially, by forwarding the mail to a UUCP feed or treating it as local. However, in some cases (such as Internet firewalls) you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes **sendmail** to try this. The downside is that errors in your configuration are likely to be diagnosed as "host unknown" or "message timed out" instead of something more meaningful. This option is not recommended.

x*LA* — When the system-load average exceeds *LA*, just queue messages (that is, do not try to send them). Defaults to 8.

X*LA* — When the system load average exceeds *LA*, refuse incoming SMTP connections. Defaults to 12.

y*fact* — The indicated *fact*or is added to the priority (thus *lowering* the priority of the job) for each recipient, that is, this value penalizes jobs with large numbers of recipients. Defaults to 30,000.

Y — If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, because the default tends to consume considerable amounts of memory while the queue is being processed.

z*fact* — The indicated *fact*or is multiplied by the message class (determined by the "Precedence:" field in the user header and the **P** lines in the configuration file) and subtracted from the priority. Thus, messages with a higher Priority are favored. Defaults to 1800.

Z*fact* — The *fact*or is added to the priority every time a job is processed. Thus, each time a job is processed, its priority is decreased by the indicated value. In most environments, this should be positive, because hosts that are down are all too often down for a long time. Defaults to 90,000.

7 — Strip input to seven bits for compatibility with old systems. This should not be necessary.

All options can be specified on the command line using the **-o** flag, but most will cause **sendmail** to relinquish its setuid permissions. The options that will not cause this are **b**, **d**, **e**, **E**, **i**, **L**, **m**, **o**, **p**, **r**, **s**, **v**, **C**, and **7**. Also, **M** (define macro) when defining the **r** or **s** macros is also considered "safe".

P—precedence definitions

Values for the "Precedence:" field may be defined using the **P** control line. The syntax of this field is:

P*name=num*

When the ***name*** is found in a "Precedence:" field, the message class is set to ***num***. Higher numbers mean higher precedence. Numbers below zero have the special property that if an error occurs during processing, the body of the message will not be returned; this is expected to be used for "bulk" mail such as through mailing lists. The default precedence is zero. An example list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Plist=-30
Pbulk=-60
Pjunk=-100
```

People writing mailing list exploders are encouraged to use "Precedence: list". Older versions of **sendmail** (which discarded all error returns for negative precedences) did not recognize this name, giving it a default precedence of zero. This allows list maintainers to see error returns on both old and new versions of **sendmail**.

V—configuration version level

To provide compatibility with old configuration files, the **V** line has been added to define some very basic semantics of the configuration file. These are not intended to be long term supports; rather, they describe compatibility features which will probably be removed in future releases.

NOTE These version "levels" have nothing to do with the version "number" on the files. For example, as of this writing version 8 configuration files (specifically, 8.6) used version level 5 configurations.

"Old" configuration files are defined as version level one.

Version level two files make the following changes:

- Host name canonification ($[... $]) appends a dot if the name is recognized; this gives the configuration file a way of finding out if anything matched. (Actually, this just initializes the "host" map with the **-a** flag—you can reset it to anything you prefer by declaring the map explicitly.) Default host name extension is consistent throughout processing; version level one configurations turned off domain extension (that is, adding the local domain name) during certain points in processing. Version level two configurations are expected to include a trailing dot to indicate that the name is already canonical.

- Local names that are not aliases are passed through a new distinguished ruleset five; this can be used to append a local relay. This behavior can be prevented by resolving the local name with an initial "@". That is, something that resolves to a local mailer and a user name of *vikki* will be passed through ruleset five, but a user name of *@vikki* will have the "@" stripped, will not be passed through ruleset five, but will otherwise be treated the same as the prior example. The expectation is that this might be used to implement a policy where mail sent to *vikki* was handled by a central hub, but mail sent to *vikki@localhost* was delivered directly.

Version level three files allow # initiated comments on all lines. Exceptions are backslash escaped # marks and the $# syntax.

Version level four configurations are completely equivalent to level three for historical reasons.

Version level five configuration files change the default definition of **$w** to be just the first component of the hostname.

K—key file declaration

Special maps can be defined using the line:

Kmapname mapclass arguments

The ***mapname*** is the handle by which this map is referenced in the rewriting rules. The ***mapclass*** is the name of a type of map; these are compiled into **sendmail** and are currently:

- btree
- dequote
- hash
- host
- implicit
- nis

The ***arguments*** are interpreted depending on the class; typically, there would be a single argument naming the file containing the map.

Maps are referenced using the syntax:

$(***map key*** $@ ***arguments*** $: ***default*** $)

where either or both of the ***arguments*** or ***default*** portion may be omitted. The ***arguments*** may appear more than once. The indicated ***key*** and ***arguments*** are passed to the appropriate mapping function. If it returns a value, it replaces the input. If it does not return a value and the ***default*** is specified, the ***default*** replaces the input. Otherwise, the input is unchanged.

During replacement of either a map value or default, the string "%*n* " (where *n* is a digit) is replaced by the corresponding ***argument***. Argument zero is always the database key. For example, the following rule looks up the UUCP name in a (user defined) UUCP map; if not found it turns it into *.UUCP* form.

```
R$- ! $+      $: $(uucp $1 $@ $2 $: %1 @ %0 . UUCP $)
```

The database might contain records like:

```
decvax          %1@%0.DEC.COM
research        %1@%0.ATT.COM
```

The built in map with both name and class "host" is the host name canonicalization lookup. Thus, the syntax:

$(host ***hostname***$)

is equivalent to:

$[***hostname***$]

There are three predefined database lookup classes:

- btree
- hash
- nis

All three accept as arguments the same optional flags and a filename (or a mapname for NIS; the filename is the root of the database path, so that ".db" or some other extension appropriate for the database type will be added to get the actual database name). Known flags are:

-o Indicates that this map is optional—that is, if it cannot be opened, no error is produced, and **sendmail** will behave as if the map existed but was empty.

-N Normally when maps are written, the trailing null byte is not included as part of the key. If this flag is indicated it will be included. During lookups, only the null-byte-included form will be searched. See also **-O**.

-O If neither **-N** or **-O** are specified, **sendmail** uses an adaptive algorithm to decide whether or not to look for null bytes on the end of keys. It starts by trying both; if it finds any key with a null byte it never tries again without a null byte and vice versa. If this flag is specified, it never tries with a null byte; this can speed matches but is never necessary. If both **-N** and **-O** are specified, **sendmail** will never try any matches at all—that is, everything will appear to fail.

-a*x* Append the string *x* on successful matches. For example, the default "host" map appends a dot on successful matches.

-f Do not fold upper to lowercase before looking up the key.

-m Match only (without replacing the value). If you only care about the existence of a key and not the value (as you might when searching the NIS map *hosts.byname* for example), this flag prevents the map from substituting the value. However, the **-a** argument is still appended on a match, and the default is still taken if the match fails.

The program **makemap**(ADMN) can be used to build any of the three database-oriented maps. It takes the following flags:

-f Fold upper to lowercase in the map.

-N Include null bytes in keys.

-o Append to an existing (old) file.

-r Allow replacement of existing keys; normally, re-inserting an existing key is an error.

-v Print what is happening.

The **sendmail** daemon does not have to be restarted to read the new maps as long as you change them in place; file locking is used so that the maps will not be read while they are being updated. (That is, do not create new maps and then use **mv**(C) to move them into place. This is a deficiency in **sendmail**.)

There are also two built-in maps that are, strictly speaking, not database look-ups.

The "host" map does host domain canonification; given a host name it calls the name server to find the canonical name for that host.

The "dequote" map strips double quotes (") from a name. It does not strip backslashes. It will not strip quotes if the resulting string would contain unscannable syntax (that is, basic errors like unbalanced angle brackets; more sophisticated errors such as unknown hosts are not checked). The intent is for use when trying to accept mail from systems such as DECnet that routinely quote odd syntax such as

```
"49ers::ubell"
```

A typical usage is probably something like:

```
Kdequote dequote

...

R$-     $: $(dequote $1 $)
R$- $+  $: $>3 $1 $2
```

Care must be taken to prevent unexpected results; for example,

```
"|someprogram < input > output"
```

will have quotes stripped, but the result is probably not what you had in mind. Fortunately these cases are rare.

Building a configuration file from scratch

Building a configuration file from scratch is an extremely difficult job. Fortunately, it is almost never necessary to do so; nearly every situation that may come up can be resolved by changing an existing configuration file. In any case, it is critical that you understand what you are trying to do and come up with a design plan for the configuration file. This section is intended to explain the real purpose of a configuration file and to give you some ideas as to what your design plan might be.

Do not even consider writing your own configuration file without carefully studying RFC 821, RFC 822, and RFC 1123. You should also read RFC 976 if you are doing UUCP exchange.

Purpose of the configuration file

The configuration file has three major purposes. The first and simplest is to set up the environment for **sendmail**. This involves setting the options, defining a few critical macros, and so on. Because these are described in other sections, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer.

The **sendmail** program performs this second phase in the following three subphases: Rulesets one and two are applied to all sender and recipient addresses, respectively. After this, you can specify per-mailer rulesets for both sender and recipient addresses. This allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose of the configuration file is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. This describes the interface requirements of the mailer.

Relevant issues

The canonical form you use should almost certainly be as specified in the Internet standards documents RFC 819 and RFC 822. These RFCs can be **ftp**'d from *ftp.ds.internic.net*.

RFC 822 describes the format of the mail message itself. The **sendmail** program 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:

< > ( ) " \

Any attempt to use these characters for other than their RFC 822 purpose in addresses is probably doomed to disaster.

RFC 819 describes the specifics of the domain-based addressing. This is touched on in RFC 822 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 need not be physical hosts; the domain is logical rather than physical. For example, at*Ocelot*, one legal host might be *a.CC.Ocelot.EDU*; reading from right to left, *EDU* is a top level domain comprising educational institutions, *Ocelot* is a logical domain name, *CC* represents the Computer Center, (in this case a strictly logical entity), and *"a"* is a host in the Computer Center.

Beware when reading RFC 819 that there are a number of errors in it.

How to proceed

Once you have decided on a design plan for the new configuration file, it is worth examining existing configuration files to decide whether any of them are close enough for you to use major parts of them as a basis for your own configuration file. Even under the worst of conditions, there should be a large amount of material you can use.

The next step is to build ruleset three. This is the hardest part of the job. Beware of doing too much to the address in this ruleset, because anything you do reflects through to the message. In particular, stripping of local domains is best deferred, as this can leave you with addresses with no domain specifications at all. Because **sendmail** likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also, try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The *Ocelot* configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively trivial. If you need hints, examine the supplied configuration files.

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

This reads the configuration file *test.cf* and enters test mode. In this mode, you enter lines of the form:

rwset address

Here ***rwset*** is the rewriting set you want to use and ***address*** is an address to which to apply the set. Test mode shows you the steps it takes as it proceeds, finally showing you the address with which it ends up. You may use a comma-separated list of rwsets for sequential application of rules to an input. For example:

3,1,21,4 monet:bollard

The first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets 21 and 4.

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

sendmail -bt -d21.99

This turns on a huge amount of information. A single-word address probably prints out several pages of information.

You should be warned that internally, **sendmail** applies ruleset 3 to all addresses. In this version of **sendmail**, you will have to do that manually. For example, older versions allowed you to use

0 bruce@broadcast.sony.com

This version requires that you use:

3,0 bruce@broadcast.sony.com

Building mailer descriptions

To add an outgoing mailer to your mail system, you must define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names "local" and "prog" must be defined.

The pathname of the mailer must be given in the "P" field. If this mailer should be accessed via an IPC connection, use the string "[IPC]" instead.

The "F" field defines the mailer flags. You should specify an "**f**" or "**r**" flag to pass the name of the sender as a **-f** or **-r** flag, respectively. These flags are only passed if they were passed to **sendmail**, so that mailers that give errors under some circumstances can be placated. If the mailer does not produce many errors, you can just specify "-f $g" in the argv template. If the mailer must be called as *root*, the **S** flag should be given. This does not reset the user ID before calling the mailer. (**sendmail** must be running setuid to root for this to work.) If this mailer is local (that is, it performs final delivery rather than another network hop), the **l** flag should be given. Quote characters (backslashes and " marks) can be stripped from addresses if the **s** flag is specified. If this is not given, they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction, the **m** flag should be stated. If this flag is on, then the argv template containing **$u** is repeated for each unique user on a given host. The **e** flag marks the mailer as being expensive, which causes **sendmail** to defer connection until a queue run. (The "c" configuration option must be given for this to be effective.)

An unusual case is the **C** flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain specification of the sender (that is, the *@host.domain* part) is saved and is appended to any addresses in the message that do not already contain a domain specification. For example:

```
From: eric@vango.CS.Berkeley.EDU
To: wnj@monet.CS.Berkeley.EDU, mckusick
```

A message of this form is modified to:

```
From: eric@vango.CS.Berkeley.EDU
To: wnj@monet.CS.Berkeley.EDU, mckusick@monet.CS.Berkeley.EDU
```

This happens if and only if the **C** flag is defined in the mailer resolved to by running *eric@vango.CS.Berkeley.EDU* through rulesets 3 and 0.

Other mailer flags are described in the section "M—define mailer" (page 200).

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

```
From: eric
```

A header of this form might be changed to be:

```
From: eric@vango.CS.Berkeley.EDU
```

Or to this form, depending on the domain it is being shipped into:

```
From: ucbvax!eric
```

These sets can also be used to do special-purpose output rewriting in cooperation with ruleset four.

The "S" and "R" fields can be specified as two numbers separated by a slash (for example, "S=10/11"), meaning that all envelope addresses will be processed through ruleset 10 and all header addresses will be processed through ruleset 11. With only one number specified, both envelope and header rewriting sets are set to the indicated ruleset.

The "E" field defines the string to use as an end-of-line indication. A string containing only newline is the default. The usual backslash escapes (\r, \n, \f, \b) may be used.

Finally, an argv template is given as the "A" field. It may have embedded spaces. If there is no argv with a **$u** macro in it, **sendmail** speaks SMTP to the mailer. If the pathname for this mailer is [IPC], the argv should be:

IPC $h [***port***]

Here ***port*** is the optional port number to connect to.

For example:

```
Mlocal, P=/usr/bin/lmail, F=lsDFMmnPS S=10, R=20, A=lmail $u
Mether, P=[IPC], F=meC, S=11, R=21, A=IPC $h, M=100000, E=\r\n
```

These specifications specify a mailer to do local delivery and a mailer for Ethernet delivery. The first is called "local"; it is located in the file */usr/bin/lmail*, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; ruleset 10 should be applied to sender addresses in the message, and ruleset 20 should be applied to recipient addresses. The argv to send to a message is the word "lmail," and words containing the name of the receiving user(s).

The second mailer is called "ether"; it should be connected to via an IPC connection; it can handle multiple users at once; connections should be deferred; and any domain from the sender address should be appended to any receiver name without a domain. Sender addresses should be processed by ruleset 11 and recipient addresses by ruleset 21. There is a 100,000-byte limit on messages passed through this mailer.

For more about sendmail

The following reference manual pages provide additional information about SCO **sendmail**:

Manual page	Description
aliases(SFF)	aliases file for **sendmail**
lmail(ADMN)	handle local mail delivery from **sendmail**
mailaddr(ADMN)	mail addressing description
mailstats(ADMN)	print statistics collected by **sendmail**
mconnect(ADMN)	connect to SMTP mail server socket
rmail(ADMN)	handle remote mail received by UUCP
sendmail(ADMN)	**sendmail** command information
sendmail.cf(SFF)	**sendmail** configuration file

Chapter 6

Administering a local MMDF system

Use this chapter for mail configuration and administration when your system uses MMDF, the Multichannel Memorandum Distribution Facility, and when you do not have networking configured.

For information on administering a networked MMDF system, or on administering mail using MMDF over UUCP dialup connections, please see Chapter 4, "Managing mail with MMDF" (page 67). If your system uses **sendmail**, see Chapter 5, "sendmail administration".

A newly installed SCO system is configured to exchange mail locally. Run the **MMDF Configuration Manager** (page 68) to perform additional MMDF configuration after installation. See also "Testing MMDF configuration" (page 77).

Once you configure MMDF, a local mail system requires minimal administration. As mail administrator for a stand-alone system, you should:

- Maintain system-wide aliases and mailing lists (page 222).
- Monitor mail queues and clean outdated mail from them periodically (page 227).
- Monitor log files in the */usr/mmdf/log* directory and remove them as necessary (page 228).
- Monitor disk space used in the */usr/spool/mail* directory where mail is received, and in the */usr/spool/mmdf* directory where queued mail is stored. See the **du**(C) and **df**(C) manual pages.

These are MMDF configuration tasks that you may want to perform at some time:

- "Adding or removing a mail user" (page 78)
- "Specifying the location of user mailboxes" (page 73).
- "Specifying the MMDF "signature"" (page 101).
- "The mmdftailor file" (page 230).

How to start a Mail manager

Start the **MMDF Configuration Manager** or any of the Mail administration managers by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, and double-clicking on the **Mail** icon. Start one of the Mail configuration or administration managers by double-clicking on its icon in the *Mail* window.

Mail administration managers are provided for:

- Hosts (page 79)
- Aliases (this page)
- Channels (page 85)
- Domains (page 94)
- Tables (page 110)

Managing mail aliases and lists

Use the **Alias Administration Manager** to:

- examine mail aliases and mailing lists and their members. The existing lists are displayed when you start the **Alias Administration Manager**. To look at the existing members, click on an alias or mailing list name (the members are displayed at the right).
- add, remove, or modify an alias (page 223).
- add or remove an alias member (page 224).
- add, remove, or modify a mailing list (page 225).
- subscribe users to aliases or mailing lists (page 226).
- retire a mail user (page 226).

Start the Alias Administration Manager by opening the *Mail* window as described in "How to start a Mail manager" (page 222) and double-clicking on the **MMDF Alias Admin** icon.

See also:

- the *mmdftailor*(F) manual page
- "Alias and mailing list tables" (page 116) for information about the tables in which MMDF maintains its system-wide alias information

Adding or removing a mail alias

To add a new alias or remove an existing one, start the **Alias Administration Manager** (this page).

To add a new alias, select **Add** from the **Aliases** menu, and specify the alias name in the "Name" field. The name may not contain ":" or "," characters. Type one or more alias member names in the "Enter Name" field and click on **Add** (or you can click on **Select** and choose each name from the list). Alias member names may be any valid user names or alias names. You must enter at least one alias member when adding a new alias.

NOTE When using an alias to define another alias, be careful not to create an alias loop, where an alias member is another alias of which the original alias is also a member (for example, alias B is a member of alias A, and alias A is a member of alias B).

Two options that are available when you select **Add** from the **Aliases** menu are not useful on a local system. See the discussions on the Allow bypass and Public options (page 82).

Maintaining alias members in a file

As an alternative to entering all alias or list members manually when adding an alias, you can maintain the list of members in a file, one member per line. When you add a new alias, in the **Enter Name** field, type:

<filename

The < indicates that the list of alias members are maintained in a file and *filename* is the location of the file. You can edit this file to modify the list of alias members.

Removing an alias

To remove an alias, start the **Alias Administration Manager** (page 223), click on the alias to be deleted, then select **Remove** from the **Aliases** menu.

See also:

- "Redirecting and piping mail" (page 225) for information on using aliases to redirect mail into files or through other programs
- the **checkaddr**(ADM) manual page for information on checking the validity of addresses
- the **list**(ADM) manual page for information on the list processor mail channel

Adding or removing an alias member

Use the **Alias Administration Manager** to add or remove new members from an existing alias.

To add alias members, start the **Alias Administration Manager** (page 223) and move the highlight to the name of the alias. Select **Add** from the **Members** menu and enter one or more names of new alias members. The member names may be any valid user names or alias names. You can type each name in the "Enter Name" field and click on **Add**, or you can click on **Select** and choose each name from the list.

To remove alias members, in the **Alias Administration Manager** (page 223) Members list, select the name or names to be deleted (press ⟨Ctrl⟩ and click to select multiple names) and then select **Remove** from the **Members** menu.

Searching an alias or mailing list

At any time while viewing or modifying the contents of an alias or mailing list, you can enter a member name or any string in the "Search" field, and the member list scrolls to that point in the alphabetically ordered list. For example, entering the letter **p** in the "Search" field will scroll the member list to those names starting with "p".

Redirecting and piping mail

When adding an alias member, you can use the forward slash (/) or the pipe character (|) to do more complex processing of mail sent to the alias, such as redirecting messages to files. For example, you can enter any of these lines as an alias "member":

```
"mmdf|cat -v >>/usr/spool/log/mlog"
"mmdf|/usr/bin/lp -dprinter2"
```

In the first example line, MMDF pipes mail addressed to the alias through the **cat**(C) filter before logging it in the *mlog* file. In the third line, MMDF pipes mail addressed to the alias to the **lp**(C) command for printing. These redirection alias examples use the user and group IDs of the user *mmdf*. You can specify any user named in the */etc/passwd* file on your system.

Maintaining mailing lists

Use the **Alias Administration Manager** (page 223) to add, modify, or remove mailing lists. A mailing list is a special-case alias; each has an "owner" who is responsible for handling mailing list administration, including adding and removing members. Error messages about bounced messages and other problems with the list also go to the mailing list owner instead of the originator of the mail. An advantage to mailing lists is that list addresses are not validated at the time each message is sent; this can save a significant amount of processing time with a large list of recipients.

To add a mailing list, add an alias (page 223), giving it the name you want to assign to the mailing list. When you have added the mailing list members, click on the **Mailing List** button near the bottom of the window and assign the mailing list owner by typing the user's login name or selecting a name from the list by clicking on **Select Name**.

The owner of the mailing list can easily maintain the list of members if it is kept in a file (page 223). The mailing list owner must have write permission on the file.

To modify a mailing list, add or remove members as described in "Adding or removing an alias member" (page 224). To change the list owner, replace the existing owner's name in the window with the new owner's name.

To remove a mailing list, follow the instructions in Removing an alias (page 224).

Adding or removing a mail user

When you add or remove a user from the system (for example, a new or retiring employee), there are several mail-related tasks that may need to be performed:

- Adding or removing the user (this page) from system-wide aliases.
- Redirecting mail (this page) sent to the retired account to one or more other users.
- Deciding what to do with a removed user's mailbox: leave it in place, remove it, or move it to another file (page 227).

When you add a new user to the system, a mailbox is automatically set up for the account in the location specified with the **MMDF Configuration Manager**, as described in "Specifying the location of user mailboxes" (page 73).

Adding or removing a user from system-wide aliases

To add a user to one or more existing system-wide aliases, start the **Alias Administration Manager** (page 223), and select **Subscribe** from the **Users** menu. Type the name of one or more users or select them from the list by clicking on the **Select** button. Select an alias by clicking on it (select multiple aliases by dragging the cursor). Click on **Add** to add the user to the selected aliases. If you change your mind about an alias, you can select it from the "Aliases user subscribes to" box and press **Remove**.

Redirecting mail to another user

When a user's account is retired, you may wish to redirect their mail to another person, such as their manager. To do this, start the **Alias Administration Manager** (page 223) and select **Retire** from the **Users** menu. Enter the name of the retiring user or click on **Select** and choose from the list. Type one or more account names to which you want to redirect the retired user's mail in the **"Redirect Mail"** field.

This removes the user's name from all system-wide aliases, and redirects their mail.

Removing or moving a retired mailbox

The mailbox from a retired account can be left in place, removed, or renamed. To do this, start the **Alias Administration Manager** (page 223) and select **Retire** from the **Users** menu. Enter the name of the retiring user or click on **Select** and choose from the list. Make your selection under "Disposition of mailbox". If you specify **Rename**, type the name of a file in which to save the mailbox.

NOTE Because of the file permissions on user mailboxes, this is the only mail administration task that must be performed as *root*. To remove or move a mailbox, log in as *root* before running the **Alias Administration Manager**.

Checking the status of mail queues

Use the **checkque**(ADM) program to check the status of the mail queues. This utility reports on the number of messages waiting for delivery. For example, when logged in as *root* or *mmdf*, enter:

/usr/mmdf/bin/checkque | more

Here is an example of output from **checkque**:

```
Thu May  9 16:07:  2 queued msgs / 80 byte queue directory
                   4 Kbytes in msg dir

  2 msgs    4 Kb (local    ) local       :  Local Delivery
                   deliver start         :  Thu May  9 15:15
                   deliver message       :  Thu May  9 15:15
                   deliver end           :  Thu May  9 15:15 / 0 hours
  *** WAITING **   First message         :  Wed Apr 24 16:03
```

If a queue is backed up with waiting mail, you can manually force **deliver**(ADM) to deliver the mail using this command:

deliver -w

The **-w** option directs **deliver** to output informative messages as it attempts to deliver the mail. See the **deliver**(ADM) manual page for additional options. You can review the output for abnormalities, such as a rejected sender or recipient. For a complete description of **checkque**, see the **checkque**(ADM) manual page.

Removing old mail from the queues

Use the **cleanque**(ADM) program to remove outdated files from the mail queues. Use **crontab**(C) as *mmdf* to run **cleanque** daily. You might want to run **cleanque** more often, depending on your mail volume. You can also run **cleanque** using **cron**(C). To do this, create a *crontabs* file for *mmdf* in the */usr/spool/cron/crontabs* directory that includes this line:

```
0    0    *    *    0-6    /usr/mmdf/bin/cleanque
```

You can also run **cleanque** by hand whenever you suspect a problem with mail delivery. The **cleanque** manual page provides a complete description of this program.

Monitoring log files

You should periodically check the size of log files in */usr/mmdf/log*. To limit the amount of log data that accumulates in a log file, move it to another filename (for example, *chan.log-*).

If you move aside a log file, make sure that *mmdf* owns the new empty file. To back up the log file and create a new empty file, enter these commands as *mmdf*:

```
cd /usr/mmdf/log
cp chan.log chan.log-
cat /dev/null > chan.log
```

MMDF configuration files

MMDF uses a variety of configuration files. Those listed here are likely to be used on a local mail system:

- mmdftailor file (page 230)

  The *mmdftailor* file defines all the MMDF mail attributes for the local computer.

- Alias tables (page 229)

  MMDF uses alias tables to obtain information about aliases and their members.

These files are all ASCII text and may be modified with a text editor. When adding long lines, you can use a backslash (\) as a line-continuation character.

When you perform the initial MMDF configuration, the **MMDF Configuration Manager** sets up the appropriate table and *mmdftailor* files.

The list.chn table

The *list.chn* table contains information about the **list-processor** program:

```
list-processor: list-processor
list-proc:      list-processor
```

These entries tell MMDF to pass mail addressed to a mailing list to the **list-processor** program. See "Alias and mailing list tables" (this page) for more information.

Alias and mailing list tables

System-wide alias and mailing list database tables are created by default in the */usr/mmdf/table* directory (or the directory specified by the **MTBLDIR** parameter in the *mmdftailor* file.)

By default, SCO systems include one channel table called *alias.n* in */usr/mmdf/table*.

On a local system, common types of alias definitions include:

- system administrative aliases for the local host, for delivery of mail to accounts such as *root*, *mmdf*, *postmaster*, and *uucp*.
- aliases for mailing lists. With list aliases, error messages and mail about problems with the list are sent to the list owner, not the sender. See "Maintaining mailing lists" (page 225).

See also:

- the *mmdftailor*(F) manual page
- "How alias tables are created" (page 117)
- "How MMDF uses alias tables" (page 118)

Assigning Mail IDs

Normally a user's login name is used as identification in a mail address. As an alternative, you can disassociate login names from mail addresses by assigning a Mail ID for each user. To do so, define the MMAILID parameter in the */usr/mmdf/mmdftailor* file to be 1.

Then map Mail IDs to login names in two table files: *users* and *mailids*. The location of these table files is defined by **MTBLDIR** in *mmdftailor*; the default is */usr/mmdf/tables*. Each table has lines with two entries per line (user and Mail ID, or Mail ID and user). The *users* file is used to map user names (login ids) to Mail IDs, for example:

```
rogerr  Roger
```

The *mailids* file is used to map Mail IDs to user names, for example:

```
Roger rogerr
```

A user can have more than one entry in the *mailids* file, but should have only one entry in the *users* file (the *users* file defines default Mail IDs).

The mmdftailor file

The */usr/mmdf/mmdftailor* file defines all the MMDF mail attributes for the local machine, such as its host name, the domain, channel, and alias tables to use, as well as how to set up each channel, and how to perform logging.

MMDF is distributed with a simple *mmdftailor* file that is configured for local mail only. When you perform the initial MMDF configuration, the **MMDF Configuration Manager** modifies the organization of the default *mmdftailor* file. If you make changes using the **MMDF Configuration Manager** or any of the MMDF administration managers, modifications are made to this and other MMDF configuration files.

You can modify *mmdftailor* and other configuration files with a text editor (page 232). However, we recommend that you use the **MMDF Configuration Manager** and the MMDF administration managers whenever possible. When you do make changes with a text editor, be sure to rebuild the hashed database (page 232) when you are finished.

This section describes some of the common MMDF keywords in *mmdftailor*. For a complete list, see the *mmdftailor*(F) manual page.

MFAILTIME is the time (in hours) a message can remain in a queue before MMDF sends a failed mail message to the sender and purges the message from the queue. Example format:

```
MFAILTIME       168
```

MLCKTYPE specifies the locking protocol MMDF uses when locking users' mailboxes. This is useful if the users on the system use third-party MUA's that use a lock file that is different from the standard UNIX System V lock file. Example format:

```
MLCKTYPE advisory
```

MMDF locking keywords	
advisory	System V **fcntl()** kernel locking protocols.
v7	Version 7 and System V Release 3, and earlier locking protocols. Creates a file called ***username.lock*** in */usr/spool/mail*. ***username*** is the name of the user's mailbox.
xenix	XENIX® system locking protocols. Creates a file called */tmp/**username**.mlk*. ***username*** is the name of the user's mailbox.
all	all of the above locking protocols (default).

The default locking protocol is **all**; however the **MMDF Configuration Manager** sets **MLCKTYPE** to **advisory**.

If you specify more than one locking protocol on the **MLCKTYPE** line, MMDF must satisfy all the locking protocols before a mailbox is considered locked. For example:

```
MLCKTYPE advisory, xenix
```

In this case, MMDF must lock the mailbox, using the **fcntl**() kernel file locking protocol *and* create a file called */usr/spool/mail/**username**.lock*. If it fails to perform both of these locks, MMDF releases the successful lock and tries again later.

MWARNTIME specifies the time (in hours) that a message can remain in a queue before MMDF sends a warning message about delayed delivery to the sender. Example format:

```
MWARNTIME       72
```

MMSGLOG controls the logging information produced by the **deliver**(ADM) and **submit**(ADM) programs. Example format:

```
MMSGLOG /tmp/mmdf/mmdfmsg.log, level=FST, size=40, stat=some
```

See the *logs*(F) manual page and the section on **MCHANLOG** in the *mmdftailor*(F) manual page.

MCHANLOG controls MMDF logging, except for information controlled by **AUTHLOG** and **MMSGLOG**. See the *mmdftailor*(F) manual page for details.

MSLEEP specifies the number of seconds that the **deliver** daemon sleeps between scanning the queues. The default is 600 (10 minutes); we recommend 60 seconds. Example format:

```
MSLEEP    60
```

Editing configuration files manually

If you modify your MMDF configuration files (page 228) manually (using a text editor), keep in mind the following guidelines:

- Always perform configuration while logged in as *mmdf*.
- Always rebuild the hashed database (this page) after modifying files.
- If you make any changes to the channel configuration (to, for example, add or remove a channel), restart the **deliver**(ADM) daemon (page 98).

See also:

- the **dbmbuild**(ADM) manual page

Rebuilding the hashed database

MMDF maintains configuration information in a hashed database for use in quickly verifying addresses and efficiently assigning channels to submitted messages. The **MMDF Configuration Manager** and the MMDF administration managers rebuild the MMDF Hashed Database automatically after modifying the configuration files.

However, if you modify */usr/mmdf/mmdftailor* or the files in the */usr/mmdf/table* directory **after** running the **MMDF Configuration Manager**, you must rebuild this database manually each time you make a modification.

To do this, enter the following commands as user *mmdf*:

```
cd /usr/mmdf/table
./dbmbuild
```

NOTE If you specify any options on the **dbmbuild**(ADM) command line, you must add the **-n** option as well. This option creates a new database instead of updating the existing one. To watch the progress of the build, enter **dbmbuild -vn**.

Mailbox locking

MMDF locks users' mailboxes while they are being modified. By default MMDF is configured to use UNIX System V **fcntl()** kernel locking protocols. You can choose to use XENIX system locking protocols, v7 protocols (Version 7 file-based locking), or all protocols. If MMDF fails to perform the specified locks, it releases any lock it has placed and tries again later.

See the section on the **MLCKTYPE** keyword in "The mmdftailor file" (page 230).

Appendix A

mail(C) commands

This appendix summarizes the commonly used commands and variables recognized by **mail**(C) as discussed in Chapter 2, "Using e-mail" (page 13).

Starting mail

Command	Description
mail	invoke **mail** program
mail -e	test for presence of mail; exit if none
mail -f myfolder	read messages from a mail folder
mail -s "report" joe < status	send file to another person

Commands available at the mail prompt

Command	Abbreviation	Description
!	**!**	execute UNIX system command
alias	**a**	create personal mailing list
Alias	**A**	find out who belongs to a mailing list
delete	**d**	delete message(s)
dp	**dp**	delete current message and print the next
edit	**e**	edit existing message
exit	**x**	exit **mail** without updating the folder
folder	**fold**	switch folders
forward	**fo**	forward message (indented) to another person
Forward	**F**	forward message (not indented)
headers	**h**	display headers of messages in the mailbox
help	**hel**	display usage of available commands
hold	**ho**	keep messages in system mailbox
list	**l**	display available commands
lpr	**lp**	send messages to the printer
mail	**m**	create message
print	**p**	display non-deleted messages
quit	**q**	update folder and quit **mail**
reply	**r**	send reply to author only
Reply	**R**	send reply to everyone in "To" list
save	**s**	save message(s) to folder
set	**se**	set **mail** variable
shell	**sh**	start subshell
undelete	**u**	restore deleted message(s)
unset	**uns**	unset **mail** variable
visual	**v**	edit existing message using **vi**
write	**w**	write message(s) to file

Commands available while composing a message

Command	Description
~!	execute a shell command
~?	list commands
~a	include your signature
~b	change "Bcc" list
~c	change "Cc" list
~d	read in *dead.letter* file
~h	change header
~m *number*	read in a message (indented)
~M *number*	read in a message (not indented)
~q	abort current message (saved as *dead.letter*)
~r *filename*	read in a file
~s	change "Subject"
~t	add names to "To" list
~v	edit message using **vi**

Variables used with the set command

Variable	Description
askcc	prompt for carbon copy recipients
asksub	prompt for "Subject" line
chron	show message headers in chronological order, most recent last
mchron	show message headers in reverse chronological order, most recent first
dot	" . " at start of line means end-of-file
folder="myfolder"	directory *myfolder* holds your mail folders
hold	keep messages in system mailbox when you quit
crt=22	page 22 lines at a time
PAGER=/usr/bin/page	display messages using **page**
prompt="mail: "	change prompt to "mail: "
quiet	stop **mail** displaying its starting message
screen=10	display 10 headers at a time
sign="That's all, folks"	assign your signature, included using **~a**
VISUAL=/usr/bin/vedit	edit messages using **vedit**

Appendix B

Mail/MMDF glossary

alias

A name that MMDF translates into a corresponding mail address or a list of mail addresses. Using an alias, you can specify a single name to represent a group of mail users. For example, using an alias called *sales* that includes and *sue* (members of the sales department), you can address mail to *"sales"* and *mike, tom, sally,* and *sue* will each get a copy.

alias loop

A state that occurs when an alias includes a member alias of which the original alias is a member (for example, alias B is a member of alias A, and alias A is a member of alias B).

attachment

A special mail item that is associated with a multipart mail message in *Using Mail.* One or more attachments may be included in a single mail message. An attachment may contain ASCII text, data from a spreadsheet or a word processor, a graphical image, or audio or video information.

badhost

A mail channel through which all mail addressed to unrecognized hosts is routed. The mail is then directed to the smart host.

baduser

A mail channel through which all mail addressed to unrecognized users is routed. The mail is then directed to the smart host.

bounced mail

Mail that is returned to the sender as undeliverable.

channel
A method that permits a machine to exchange data using a single type of network communications protocol. It handles the mail transport protocol so that neither the operating system nor the rest of the MMDF system has to know about the intricacies of a particular mail transport protocol. Channels act not only as protocol handlers, but in some cases actually initiate the communications to the network or to another machine as needed. They also may convert address or message formats as necessary.

channel input program
The program used by a channel to monitor and manage incoming mail. For example, */usr/bin/rmail* is used by the UUCP channel.

channel output program
The program used by a channel to process outgoing mail. For example, */usr/mmdf/chans/uucp* is used by the UUCP channel.

channel queue
The holding space used by a channel for incoming or outgoing mail that is waiting to be delivered.

domain name
A name used for mail delivery that describes the site where a computer is located and generally includes the machine (host) name, a department (optionally), and the site's organization or country.

domain name server
A computer that maintains a database that can provide information about host names and addresses within a domain. See name server (page 241).

domain tables
Tables that are used to match a short host name to its fully qualified host name. Domain tables can be used to convey information to MMDF about how machines are connected and about any special domain routing considerations.

fully qualified domain name
The full name of a host, by which it can be uniquely identified. The name includes the machine name and the full name for the domain to which it belongs. Also called the fully qualified host name.

fully qualified host name
Same as a fully qualified domain name (this page).

hashed database
MMDF configuration information built by **dbmbuild**(ADM) and maintained for quick verification of addresses and efficiency in assigning channels to submitted messages. The hashed database must be rebuilt after manual modification of MMDF configuration files.

headers

Information included at the top of every MMDF mail message. Headers minimally identify the message sender, recipient, and the date it was sent, and can optionally provide other information such as carbon copy recipients, message ID information, and message priority.

host

Any computer that supports exchange of mail over MMDF. See local host (this page).

Host name

Sometimes called a system name or machine name. The name of the machine on which you are configuring MMDF. To determine the local host name, enter **uname -n** at the UNIX system prompt. An example of a host name is *scribe*.

MMDF allows host names to contain underscore characters; this is an extension of internet standards. To ensure interoperability with hosts that strictly conform to internet standards, SCO recommends that you do not use the underscore character when internetworking is a primary goal.

local host

The machine on which you are currently managing MMDF.

mailing list

A special type of alias. Each mailing list has an "owner" who handles list administration and receives any error messages about bounced messages or other problems with list mail. Mailing list addresses are not validated at the time each message is sent, which can save a significant amount of processing time with a large list of recipients.

Mail Transfer Agent (MTA)

A facility that accepts messages generated by an Mail User Agent, determines a route for delivery, edits the message header as required by the destination and delivery program, and calls the appropriate delivery program to deliver the mail. SCO systems provide two MTAs: MMDF and **sendmail**(ADMN).

Mail User Agent (MUA)

A program that provides a user interface for mail message processing, including mail composition and reading. SCO systems provide three MUAs: **mail**(C), **scomail**(XC), and **SCO Shell Mail** (page 39).

MIME

Multipurpose Internet Mail Extensions. A standard for mail exchange that supports graphical, audio, video, and multimedia messages. MMDF is MIME compliant.

MMDF

Multichannel Memorandum Distribution Facility. One of the two Mail Transfer Agents used on SCO systems. The other is sendmail (page 149). MMDF provides transparent access to the different networks and mail transport protocols, regardless of the Mail User Agent employed (one or more MUAs may be used on a single system).

MMDF Alias Administration Manager

A program that provides an interactive interface for accomplishing daily administration of your mail aliases.

Start the **Alias Administration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, and double-clicking on the **MMDF Alias Admin** icon.

MMDF Channel Administration manager

A program that provides an interactive interface for accomplishing advanced administration of your mail channels.

Start the **Channel Administration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, double-clicking on the **MMDF_Advanced_Admin** icon, and double-clicking on the **MMDF Channel Admin** icon.

MMDF Configuration manager

A program used for configuring MMDF on the local host. This program may be run at installation time and any time after that when the system undergoes a major change, such as when you connect your machine to a new network or remove it from an existing one.

Start the **MMDF Configuration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, and double-clicking on the **MMDF Configuration** icon. Alternatively, enter **mkdev mmdf** at a command line.

MMDF Domain Administration manager

A program that provides an interactive interface for accomplishing advanced administration of your mail domains.

Start the **Domain Administration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, double-clicking on the **MMDF_Advanced_Admin** icon, and double-clicking on the **MMDF Domain Admin** icon.

MMDF Host Administration manager

A program that provides an interactive interface for accomplishing daily administration of your mail hosts.

Start the **Host Administration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, and double-clicking on the **MMDF Host Admin** icon.

MMDF Table Administration manager

A program that provides an interactive interface for accomplishing advanced administration of your mail tables.

Start the **Table Administration Manager** by selecting **Toolsheds** from the Desktop **Tools** menu, double-clicking on the **System_Administration Tools** icon, double-clicking on the **Mail** icon, double-clicking on the **MMDF_Advanced_Admin** icon, and double-clicking on the **MMDF Table Admin** icon.

name server

A program running on a network that provides a central database of information, such as Internet addresses and the names of hosts on which people receive mail.

postmaster

An address that is required for every domain on the Internet by RFC 821/822. People send inquiries about user and host names to the *postmaster* address in that domain.

RFC 822

Internet Technical Bulletin, available from the InterNIC Registration Services, or InterNIC. See "Registering domain names" (page 97) for more information.

smart host

A computer that has more complete information about the entire mail network than do local hosts. Mail addressed to hosts or users unknown to a local host can be forwarded to the smart host over the badhosts or badusers channels.

SMTP

Simple Mail Transfer Protocol. The mail transfer protocol used over TCP/IP and the Internet.

smtp

An MMDF channel used to exchange messages over TCP/IP using the Simple Mail Transfer Protocol.

subdomain

A registered name that describes a company, department, or any subgroup under a top-level domain name (page 238). *sco* is an example of a subdomain in the domain *COM*.

A

B

C

D

E

M

N

O

P

Q

R

S

T

U

W

X

15 May 1998
AU20005P002