# SENDMAIL™

## INSTALLATION AND OPERATION GUIDE

Eric Allman
Claus Assmann
Gregory Neil Shapiro
Sendmail, Inc.

Version 8.751

For Sendmail Version 8.14

*Sendmail*™ implements a general purpose internetwork mail routing facility under the UNIX® operating system. It is not tied to any one transport protocol — its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file.

Due to the requirements of flexibility for *sendmail*, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration file incrementally.

*Sendmail* is based on RFC 821 (Simple Mail Transport Protocol), RFC 822 (Internet Mail Headers Format), RFC 974 (MX routing), RFC 1123 (Internet Host Requirements), RFC 1413 (Identification server), RFC 1652 (SMTP 8BITMIME Extension), RFC 1869 (SMTP Service Extensions), RFC 1870 (SMTP SIZE Extension), RFC 1891 (SMTP Delivery Status Notifications), RFC 1892 (Multipart/Report), RFC 1893 (Enhanced Mail System Status Codes), RFC 1894 (Delivery Status Notifications), RFC 1985 (SMTP Service Extension for Remote Message Queue Starting), RFC 2033 (Local Message Transmission Protocol), RFC 2034 (SMTP Service Extension for Returning Enhanced Error Codes), RFC 2045 (MIME), RFC 2476 (Message Submission), RFC 2487 (SMTP Service Extension for Secure SMTP over TLS), RFC 2554 (SMTP Service Extension for Authentication), RFC 2821 (Simple Mail Transfer Protocol), RFC 2822 (Internet Message Format), RFC 2852 (Deliver By SMTP Service Extension), and RFC 2920 (SMTP Service Extension for Command Pipelining). However, since *sendmail* is designed to work in a wider world, in many cases it can be configured to exceed these protocols. These cases are described herein.

Although *sendmail* is intended to run without the need for monitoring, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic *sendmail* installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install *sendmail* and keep it happy. Section three has information regarding the command line arguments. Section four describes some parameters that may

---

**DISCLAIMER:** This documentation is under modification.

be safely tweaked. Section five contains the nitty-gritty information about the configuration file. This section is for masochists and people who must write their own configuration file. Section six describes configuration that can be done at compile time. The appendixes give a brief but detailed explanation of a number of features not described in the rest of the paper.

# 1. BASIC INSTALLATION

There are two basic steps to installing *sendmail*. First, you have to compile and install the binary. If *sendmail* has already been ported to your operating system that should be simple. Second, you must build a run-time configuration file. This is a file that *sendmail* reads when it starts up that describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration file can be quite complex, a configuration can usually be built using an M4-based configuration language. Assuming you have the standard *sendmail* distribution, see *cf/README* for further information.

The remainder of this section will describe the installation of *sendmail* assuming you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the *sendmail* subtree, normally */usr/src/usr.sbin/sendmail* on 4.4BSD-based systems.

Continue with the next section if you need/want to compile *sendmail* yourself. If you have a running binary already on your system, you should probably skip to section 1.2.

## 1.1. Compiling Sendmail

All *sendmail* source is in the *sendmail* subdirectory. To compile sendmail, “cd” into the *sendmail* directory and type

```
./Build
```

This will leave the binary in an appropriately named subdirectory, e.g., obj.BSD-OS.2.1.i386. It works for multiple object versions compiled out of the same directory.

### 1.1.1. Tweaking the Build Invocation

You can give parameters on the *Build* command. In most cases these are only used when the *obj.** directory is first created. To restart from scratch, use *-c*. These commands include:

–L *libdirs*
: A list of directories to search for libraries.

–I *incdirs*
: A list of directories to search for include files.

–E *envar*=*value*
: Set an environment variable to an indicated *value* before compiling.

–c
: Create a new *obj.** tree before running.

–f *siteconfig*
: Read the indicated site configuration file. If this parameter is not specified, *Build* includes *all* of the files *$BUILDTOOLS/Site/site.$oscf.m4* and *$BUILDTOOLS/Site/site.config.m4*, where $BUILDTOOLS is normally *../devtools* and $oscf is the same name as used on the *obj.** directory. See below for a description of the site configuration file.

–S
: Skip auto-configuration. *Build* will avoid auto-detecting libraries if this is set. All libraries and map definitions must be specified in the site configuration file.

Most other parameters are passed to the *make* program; for details see *$BUILDTOOLS/README*.

### 1.1.2. Creating a Site Configuration File

(This section is not yet complete. For now, see the file devtools/README for details.) See sendmail/README for various compilation flags that can be set.

#### 1.1.3. Tweaking the Makefile

*Sendmail* supports two different formats for the local (on disk) version of databases, notably the *aliases* database. At least one of these should be defined if at all possible.

NDBM — The ‘‘new DBM’’ format, available on nearly all systems around today. This was the preferred format prior to 4.4BSD. It allows such complex things as multiple databases and closing a currently open database.

NEWDB — The Berkeley DB package. If you have this, use it. It allows long records, multiple open databases, real in-memory caching, and so forth. You can define this in conjunction with NDBM; if you do, old alias databases are read, but when a new database is created it will be in NEWDB format. As a nasty hack, if you have NEWDB, NDBM, and NIS defined, and if the alias file name includes the substring “/yp/”, *sendmail* will create both new and old versions of the alias file during a *newalias* command. This is required because the Sun NIS/YP system reads the DBM version of the alias file. It’s ugly as sin, but it works.

If neither of these are defined, *sendmail* reads the alias file into memory on every invocation. This can be slow and should be avoided. There are also several methods for remote database access:

LDAP — Lightweight Directory Access Protocol.

NIS — Sun’s Network Information Services (formerly YP).

NISPLUS — Sun’s NIS+ services.

NETINFO — NeXT’s NetInfo service.

HESIOD — Hesiod service (from Athena).

Other compilation flags are set in *conf.h* and should be predefined for you unless you are porting to a new environment. For more options see *sendmail/README*.

#### 1.1.4. Compilation and installation

After making the local system configuration described above, You should be able to compile and install the system. The script “Build” is the best approach on most systems:

```
./Build
```

This will use *uname*(1) to create a custom Makefile for your environment.

If you are installing in the standard places, you should be able to install using

```
./Build install
```

This should install the binary in /usr/sbin and create links from /usr/bin/newaliases and /usr/bin/mailq to /usr/sbin/sendmail. On most systems it will also format and install man pages. Notice: as of version 8.12 *sendmail* will no longer be installed set-user-ID root by default. If you really want to use the old method, you can specify it as target:

```
./Build install-set-user-id
```

### 1.2. Configuration Files

*Sendmail* cannot operate without a configuration file. The configuration defines the mail delivery mechanisms understood at this site, how to access them, how to forward email to remote mail systems, and a number of tuning parameters. This configuration file is detailed in the later portion of this document.

The *sendmail* configuration can be daunting at first. The world is complex, and the mail configuration reflects that. The distribution includes an m4-based configuration package that hides a lot of the complexity. See *cf/README* for details.

Our configuration files are processed by *m4* to facilitate local customization; the directory *cf* of the *sendmail* distribution directory contains the source files. This directory contains several subdirectories:

| | |
|---|---|
| cf | Both site-dependent and site-independent descriptions of hosts. These can be literal host names (e.g., "ucbvax.mc") when the hosts are gateways or more general descriptions (such as "generic-solaris2.mc" as a general description of an SMTP-connected host running Solaris 2.x. Files ending **.mc** (''M4 Configuration'') are the input descriptions; the output is in the corresponding **.cf** file. The general structure of these files is described below. |
| domain | Site-dependent subdomain descriptions. These are tied to the way your organization wants to do addressing. For example, **domain/CS.Berkeley.EDU.m4** is our description for hosts in the CS.Berkeley.EDU subdomain. These are referenced using the DOMAIN **m4** macro in the **.mc** file. |
| feature | Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE **m4** macro. An example feature is use_cw_file (which tells *sendmail* to read an /etc/mail/local-host-names file on startup to find the set of local names). |
| hack | Local hacks, referenced using the HACK **m4** macro. Try to avoid these. The point of having them here is to make it clear that they smell. |
| m4 | Site-independent *m4*(1) include files that have information common to all configuration files. This can be thought of as a "#include" directory. |
| mailer | Definitions of mailers, referenced using the MAILER **m4** macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use "MAILER(uucp)". |
| ostype | Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE **m4** macro. |
| sh | Shell files used by the **m4** build process. You shouldn't have to mess with these. |
| siteconfig | Local UUCP connectivity information. This directory has been supplanted by the mailertable feature; any new configurations should use that feature to do UUCP (and other) routing. The use of this directory is deprecated. |

If you are in a new domain (e.g., a company), you will probably want to create a cf/domain file for your domain. This consists primarily of relay definitions and features you want enabled site-wide: for example, Berkeley's domain definition defines relays for BitNET and UUCP. These are specific to Berkeley, and should be fully-qualified internet-style domain names. Please check to make certain they are reasonable for your domain.

Subdomains at Berkeley are also represented in the cf/domain directory. For example, the domain CS.Berkeley.EDU is the Computer Science subdomain, EECS.Berkeley.EDU is the Electrical Engineering and Computer Sciences subdomain, and S2K.Berkeley.EDU is the Sequoia 2000 subdomain. You will probably have to add an entry to this directory to be appropriate for your domain.

You will have to use or create **.mc** files in the *cf/cf* subdirectory for your hosts. This is detailed in the cf/README file.

### 1.3. Details of Installation Files

This subsection describes the files that comprise the *sendmail* installation.

#### 1.3.1. /usr/sbin/sendmail

The binary for *sendmail* is located in /usr/sbin[1]. It should be set-group-ID smmsp as described in sendmail/SECURITY. For security reasons, /, /usr, and /usr/sbin should be owned by root, mode 0755[2].

#### 1.3.2. /etc/mail/sendmail.cf

This is the main configuration file for *sendmail*[3]. This is one of the two non-library file names compiled into *sendmail*[4], the other is /etc/mail/submit.cf.

The configuration file is normally created using the distribution files described above. If you have a particularly unusual system configuration you may need to create a special version. The format of this file is detailed in later sections of this document.

#### 1.3.3. /etc/mail/submit.cf

This is the configuration file for *sendmail* when it is used for initial mail submission, in which case it is also called ‘‘Mail Submission Program’’ (MSP) in contrast to ‘‘Mail Transfer Agent’’ (MTA). Starting with version 8.12, *sendmail* uses one of two different configuration files based on its operation mode (or the new **−A** option). For initial mail submission, i.e., if one of the options **−bm** (default), **−bs**, or **−t** is specified, submit.cf is used (if available), for other operations sendmail.cf is used. Details can be found in *sendmail/SECURITY*. submit.cf is shipped with sendmail (in cf/cf/) and is installed by default. If changes to the configuration need to be made, start with cf/cf/submit.mc and follow the instruction in cf/README.

#### 1.3.4. /usr/bin/newaliases

The *newaliases* command should just be a link to *sendmail*:

```
rm −f /usr/bin/newaliases
ln −s /usr/sbin/sendmail /usr/bin/newaliases
```

This can be installed in whatever search path you prefer for your system.

#### 1.3.5. /usr/bin/hoststat

The *hoststat* command should just be a link to *sendmail*, in a fashion similar to *newaliases*. This command lists the status of the last mail transaction with all remote hosts. The **−v** flag will prevent the status display from being truncated. It functions only when the **HostStatusDirectory** option is set.

#### 1.3.6. /usr/bin/purgestat

This command is also a link to *sendmail*. It flushes expired (Timeout.hoststatus) information that is stored in the **HostStatusDirectory** tree.

---

[1]This is usually /usr/sbin on 4.4BSD and newer systems; many systems install it in /usr/lib. I understand it is in /usr/ucblib on System V Release 4.

[2]Some vendors ship them owned by bin; this creates a security hole that is not actually related to *sendmail*. Other important directories that should have restrictive ownerships and permissions are /bin, /usr/bin, /etc, /etc/mail, /usr/etc, /lib, and /usr/lib.

[3]Actually, the pathname varies depending on the operating system; /etc/mail is the preferred directory. Some older systems install it in **/usr/lib/sendmail.cf**, and I've also seen it in **/usr/ucblib**. If you want to move this file, add -D_PATH_SENDMAILCF=\"/file/name\" to the flags passed to the C compiler. Moving this file is not recommended: other programs and scripts know of this location.

[4]The system libraries can reference other files; in particular, system library subroutines that *sendmail* calls probably reference */etc/passwd* and */etc/resolv.conf*.

### 1.3.7. /var/spool/mqueue

The directory */var/spool/mqueue* should be created to hold the mail queue. This directory should be mode 0700 and owned by root.

The actual path of this directory is defined by the **QueueDirectory** option of the *sendmail.cf* file. To use multiple queues, supply a value ending with an asterisk. For example, */var/spool/mqueue/qd** will use all of the directories or symbolic links to directories beginning with 'qd' in */var/spool/mqueue* as queue directories. Do not change the queue directory structure while sendmail is running.

If these directories have subdirectories or symbolic links to directories named 'qf', 'df', and 'xf', then these will be used for the different queue file types. That is, the data files are stored in the 'df' subdirectory, the transcript files are stored in the 'xf' subdirectory, and all others are stored in the 'qf' subdirectory.

If shared memory support is compiled in, *sendmail* stores the available diskspace in a shared memory segment to make the values readily available to all children without incurring system overhead. In this case, only the daemon updates the data; i.e., the sendmail daemon creates the shared memory segment and deletes it if it is terminated. To use this, *sendmail* must have been compiled with support for shared memory (-DSM_CONF_SHM) and the option **SharedMemoryKey** must be set. Notice: do not use the same key for *sendmail* invocations with different queue directories or different queue group declarations. Access to shared memory is not controlled by locks, i.e., there is a race condition when data in the shared memory is updated. However, since operation of *sendmail* does not rely on the data in the shared memory, this does not negatively influence the behavior.

### 1.3.8. /var/spool/clientmqueue

The directory */var/spool/clientmqueue* should be created to hold the mail queue. This directory should be mode 0770 and owned by user smmsp, group smmsp.

The actual path of this directory is defined by the **QueueDirectory** option of the *submit.cf* file.

### 1.3.9. /var/spool/mqueue/.hoststat

This is a typical value for the **HostStatusDirectory** option, containing one file per host that this sendmail has chatted with recently. It is normally a subdirectory of *mqueue*.

### 1.3.10. /etc/mail/aliases*

The system aliases are held in "/etc/mail/aliases". A sample is given in "sendmail/aliases" which includes some aliases which *must* be defined:

> cp sendmail/aliases /etc/mail/aliases
> *edit /etc/mail/aliases*

You should extend this file with any aliases that are apropos to your system.

Normally *sendmail* looks at a database version of the files, stored either in "/etc/mail/aliases.dir" and "/etc/mail/aliases.pag" or "/etc/mail/aliases.db" depending on which database package you are using. The actual path of this file is defined in the **AliasFile** option of the *sendmail.cf* file.

The permissions of the alias file and the database versions should be 0640 to prevent local denial of service attacks as explained in the top level **README** in the sendmail distribution. If the permissions 0640 are used, be sure that only trusted users belong to the group assigned to those files. Otherwise, files should not even be group readable.

#### 1.3.11. /etc/rc or /etc/init.d/sendmail

It will be necessary to start up the *sendmail* daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to insure that mail gets delivered when hosts come up.

If necessary, add the following lines to "/etc/rc" (or "/etc/rc.local" as appropriate) in the area where it is starting up the daemons on a BSD-base system, or on a System-V-based system in one of the startup files, typically "/etc/init.d/sendmail":

```
if [ −f /usr/sbin/sendmail −a −f /etc/mail/sendmail.cf ]; then
        (cd /var/spool/mqueue; rm −f xf*)
        /usr/sbin/sendmail −bd −q30m &
        echo −n ' sendmail' >/dev/console
fi
```

The "cd" and "rm" commands insure that all transcript files have been removed; extraneous transcript files may be left around if the system goes down in the middle of processing a message. The line that actually invokes *sendmail* has two flags: "−bd" causes it to listen on the SMTP port, and "−q30m" causes it to run the queue every half hour.

Some people use a more complex startup script, removing zero length qf/hf/Qf files and df files for which there is no qf/hf/Qf file. Note this is not advisable. For example, see Figure 1 for an example of a complex script which does this clean up.

#### 1.3.12. /etc/mail/helpfile

This is the help file used by the SMTP **HELP** command. It should be copied from "sendmail/helpfile":

```
cp sendmail/helpfile /etc/mail/helpfile
```

The actual path of this file is defined in the **HelpFile** option of the *sendmail.cf* file.

#### 1.3.13. /etc/mail/statistics

If you wish to collect statistics about your mail traffic, you should create the file "/etc/mail/statistics":

```
cp /dev/null /etc/mail/statistics
chmod 0600 /etc/mail/statistics
```

This file does not grow. It is printed with the program "mailstats/mailstats.c." The actual path of this file is defined in the **S** option of the *sendmail.cf* file.

#### 1.3.14. /usr/bin/mailq

If *sendmail* is invoked as "mailq," it will simulate the **−bp** flag (i.e., *sendmail* will print the contents of the mail queue; see below). This should be a link to /usr/sbin/sendmail.

#### 1.3.15. sendmail.pid

*sendmail* stores its current pid in the file specified by the **PidFile** option (default is _PATH_SENDMAILPID). *sendmail* uses **TempFileMode** (which defaults to 0600) as the permissions of that file to prevent local denial of service attacks as explained in the top level **README** in the sendmail distribution. If the file already exists, then it might be necessary to change the permissions accordingly, e.g.,

```
chmod 0600 /var/run/sendmail.pid
```

Note that as of version 8.13, this file is unlinked when *sendmail* exits. As a result of this change, a script such as the following, which may have worked prior to 8.13, will no longer

```
#!/bin/sh
# remove zero length qf/hf/Qf files
for qffile in qf* hf* Qf*
do
	if [ –r $qffile ]
	then
		if [ ! –s $qffile ]
		then
			echo –n " <zero: $qffile>" > /dev/console
			rm –f $qffile
		fi
	fi
done
# rename tf files to be qf if the qf does not exist
for tffile in tf*
do
	qffile=‘echo $tffile | sed ’s/t/q/’‘
	if [ –r $tffile –a ! –f $qffile ]
	then
		echo –n " <recovering: $tffile>" > /dev/console
		mv $tffile $qffile
	else
		if [ –f $tffile ]
		then
			echo –n " <extra: $tffile>" > /dev/console
			rm –f $tffile
		fi
	fi
done
# remove df files with no corresponding qf/hf/Qf files
for dffile in df*
do
	qffile=‘echo $dffile | sed ’s/d/q/’‘
	hffile=‘echo $dffile | sed ’s/d/h/’‘
	Qffile=‘echo $dffile | sed ’s/d/Q/’‘
	if [ –r $dffile –a ! –f $qffile –a ! –f $hffile –a ! –f $Qffile ]
	then
		echo –n " <incomplete: $dffile>" > /dev/console
		mv $dffile ‘echo $dffile | sed ’s/d/D/’‘
	fi
done
# announce files that have been saved during disaster recovery
for xffile in [A-Z]f*
do
	if [ –f $xffile ]
	then
		echo –n " <panic: $xffile>" > /dev/console
	fi
done
```

Figure 1 — A complex startup script

---

work:

```
# stop & start sendmail
PIDFILE=/var/run/sendmail.pid
kill `head -1 $PIDFILE`
`tail -1 $PIDFILE`
```

because it assumes that the pidfile will still exist even after killing the process to which it refers. Below is a script which will work correctly on both newer and older versions:

```
# stop & start sendmail
PIDFILE=/var/run/sendmail.pid
pid=`head -1 $PIDFILE`
cmd=`tail -1 $PIDFILE`
kill $pid
$cmd
```

This is just an example script, it does not perform any error checks, e.g., whether the pidfile exists at all.

#### 1.3.16. Map Files

To prevent local denial of service attacks as explained in the top level **README** in the sendmail distribution, the permissions of map files created by *makemap* should be 0640. The use of 0640 implies that only trusted users belong to the group assigned to those files. If those files already exist, then it might be necessary to change the permissions accordingly, e.g.,

```
cd /etc/mail
chmod 0640 *.db *.pag *.dir
```

## 2. NORMAL OPERATIONS

### 2.1. The System Log

The system log is supported by the *syslogd*(8) program. All messages from *sendmail* are logged under the LOG_MAIL facility[5].

#### 2.1.1. Format

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 local area network), the word "sendmail:", and a message[6]. Most messages are a sequence of *name*=*value* pairs.

The two most common lines are logged when a message is processed. The first logs the receipt of a message; there will be exactly one of these per message. Some fields may be omitted if they do not contain interesting information. Fields are:

| | |
|---|---|
| from | The envelope sender address. |
| size | The size of the message in bytes. |

---

[5]Except on Ultrix, which does not support facilities in the syslog.

[6]This format may vary slightly if your vendor has changed the syntax.

| | |
|---|---|
| class | The class (i.e., numeric precedence) of the message. |
| pri | The initial message priority (used for queue sorting). |
| nrcpts | The number of envelope recipients for this message (after aliasing and forwarding). |
| msgid | The message id of the message (from the header). |
| bodytype | The message body type (7BIT or 8BITMIME), as determined from the envelope. |
| proto | The protocol used to receive this message (e.g., ESMTP or UUCP) |
| daemon | The daemon name from the **DaemonPortOptions** setting. |
| relay | The machine from which it was received. |

There is also one line logged per delivery attempt (so there can be several per message if delivery is deferred or there are multiple recipients). Fields are:

| | |
|---|---|
| to | A comma-separated list of the recipients to this mailer. |
| ctladdr | The ‘‘controlling user’’, that is, the name of the user whose credentials we use for delivery. |
| delay | The total delay between the time this message was received and the current delivery attempt. |
| xdelay | The amount of time needed in this delivery attempt (normally indicative of the speed of the connection). |
| mailer | The name of the mailer used to deliver to this recipient. |
| relay | The name of the host that actually accepted (or rejected) this recipient. |
| dsn | The enhanced error code (RFC 2034) if available. |
| stat | The delivery status. |

Not all fields are present in all messages; for example, the relay is usually not listed for local deliveries.

#### 2.1.2. Levels

If you have *syslogd* (8) or an equivalent installed, you will be able to do logging. There is a large amount of information that 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 ten are considered generally “useful;” log levels above 64 are reserved for debugging purposes. Levels from 11−64 are reserved for verbose information that some sites might want.

A complete description of the log levels is given in section ‘‘Log Level’’.

### 2.2. 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.

### 2.3. The Mail Queues

Mail messages may either be delivered immediately or be held for later delivery. Held messages are placed into a holding directory called a mail queue.

A mail message may be queued for these reasons:

- If a mail message is temporarily undeliverable, it is queued and delivery is attempted later. If the message is addressed to multiple recipients, it is queued only for those recipients to whom

delivery is not immediately possible.
- If the SuperSafe option is set to true, all mail messages are queued while delivery is attempted.
- If the DeliveryMode option is set to queue-only or defer, all mail is queued, and no immediate delivery is attempted.
- If the load average becomes higher than the value of the QueueLA option and the **QueueFactor** (**q**) option divided by the difference in the current load average and the **QueueLA** option plus one is less than the priority of the message, messages are queued rather than immediately delivered.
- One or more addresses are marked as expensive and delivery is postponed until the next queue run or one or more address are marked as held via mailer which uses the hold mailer flag.
- The mail message has been marked as quarantined via a mail filter or rulesets.
-

### 2.3.1. Queue Groups and Queue Directories

There are one or more mail queues. Each mail queue belongs to a queue group. There is always a default queue group that is called “mqueue” (which is where messages go by default unless otherwise specified). The directory or directories which comprise the default queue group are specified by the QueueDirectory option. There are zero or more additional named queue groups declared using the **Q** command in the configuration file.

By default, a queued message is placed in the queue group associated with the first recipient in the recipient list. A recipient address is mapped to a queue group as follows. First, if there is a ruleset called “queuegroup”, and if this ruleset maps the address to a queue group name, then that queue group is chosen. That is, the argument for the ruleset is the recipient address and the result should be **$#** followed by the name of a queue group. Otherwise, if the mailer associated with the address specifies a queue group, then that queue group is chosen. Otherwise, the default queue group is chosen.

A message with multiple recipients will be split if different queue groups are chosen by the mapping of recipients to queue groups.

When a message is placed in a queue group, and the queue group has more than one queue, a queue is selected randomly.

If a message with multiple recipients is placed into a queue group with the ’r’ option (maximum number of recipients per message) set to a positive value *N*, and if there are more than *N* recipients in the message, then the message will be split into multiple messages, each of which have at most *N* recipients.

Notice: if multiple queue groups are used, do **not** move queue files around, e.g., into a different queue directory. This may have weird effects and can cause mail not to be delivered. Queue files and directories should be treated as opaque and should not be manipulated directly.

### 2.3.2. Queue Runs

*sendmail* has two different ways to process the queue(s). The first one is to start queue runners after certain intervals (“normal” queue runners), the second one is to keep queue runner processes around (“persistent” queue runners). How to select either of these types is discussed in the appendix “COMMAND LINE FLAGS”. Persistent queue runners have the advantage that no new processes need to be spawned at certain intervals; they just sleep for a specified time after they finished a queue run. Another advantage of persistent queue runners is that only one process belonging to a workgroup (a workgroup is a set of queue groups) collects the data for a queue run and then multiple queue runner may go ahead using that data. This can significantly reduce the disk I/O necessary to read the queue files compared to starting multiple queue runners directly. Their disadvantage is that a new queue run is only started after all queue runners belonging to a group finished their tasks. In case one of the queue runners tries delivery to a slow recipient site at the end of a queue run, the next queue run may be substantially delayed.

In general this should be smoothed out due to the distribution of those slow jobs, however, for sites with small number of queue entries this might introduce noticable delays. In general, persistent queue runners are only useful for sites with big queues.

#### 2.3.3. Manual Intervention

Under normal conditions the mail queue will 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. In that case you want to check the content of the queue and manipulate it as explained in the next two sections.

#### 2.3.4. Printing the queue

The contents of the queue(s) can be printed using the *mailq* command (or by specifying the **−bp** flag to *sendmail*):

```
mailq
```

This will produce a listing of the queue id's, the size of the message, the date the message entered the queue, and the sender and recipients. If shared memory support is compiled in, the flag **−bP** can be used to print the number of entries in the queue(s), provided a process updates the data. However, as explained earlier, the output might be slightly wrong, since access to the shared memory is not locked. For example, ''unknown number of entries'' might be shown. The internal counters are updated after each queue run to the correct value again.

#### 2.3.5. Forcing the queue

*Sendmail* should run the queue automatically at intervals. When using multiple queues, a separate process will by default be created to run each of the queues unless the queue run is initiated by a user with the verbose flag. The algorithm is to read and sort the queue, and 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 insure that only one queue processor exists at any time, since 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 entire 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 solve this.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in *sendmail* spending 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 /var/spool
mv mqueue omqueue; mkdir mqueue; chmod 0700 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, issue the following command:

```
/usr/sbin/sendmail −C /etc/mail/queue.cf −q
```

The **−C** flag specifies an alternate configuration file **queue.cf** which should refer to the moved queue directory

O QueueDirectory=/var/spool/omqueue

and the **−q** flag says to just run every job in the queue. You can also specify the moved queue directory on the command line

/usr/sbin/sendmail −oQ/var/spool/omqueue −q

but this requires that you do not have queue groups in the configuration file, because those are not subdirectories of the moved directory. See the section about ‘‘Queue Group Declaration’’ for details; you most likely need a different configuration file to correctly deal with this problem. However, a proper configuration of queue groups should avoid filling up queue directories, so you shouldn’t run into this problem. If you have a tendency toward voyeurism, you can use the **−v** flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

rmdir /var/spool/omqueue

#### 2.3.6. Quarantined Queue Items

It is possible to "quarantine" mail messages, otherwise known as envelopes. Envelopes (queue files) are stored but not considered for delivery or display unless the "quarantine" state of the envelope is undone or delivery or display of quarantined items is requested. Quarantined messages are tagged by using a different name for the queue file, ’hf’ instead of ’qf’, and by adding the quarantine reason to the queue file.

Delivery or display of quarantined items can be requested using the **−qQ** flag to *sendmail* or *mailq*. Additionally, messages already in the queue can be quarantined or unquarantined using the new **−Q** flag to sendmail. For example,

sendmail -Qreason -q[!][I|R|S][matchstring]

Quarantines the normal queue items matching the criteria specified by the **-q[!][I|R|S][matchstring]** using the reason given on the **−Q** flag. Likewise,

sendmail -qQ -Q[reason] -q[!][I|R|S|Q][matchstring]

Change the quarantine reason for the quarantined items matching the criteria specified by the **-q[!][I|R|S|Q][matchstring]** using the reason given on the **−Q** flag. If there is no reason, unquarantine the matching items and make them normal queue items. Note that the **−qQ** flag tells sendmail to operate on quarantined items instead of normal items.

### 2.4. Disk Based Connection Information

*Sendmail* stores a large amount of information about each remote system it has connected to in memory. It is possible to preserve some of this information on disk as well, by using the **HostStatusDirectory** option, so that it may be shared between several invocations of *sendmail*. This allows mail to be queued immediately or skipped during a queue run if there has been a recent failure in connecting to a remote machine. Note: information about a remote system is stored in a file whose pathname consists of the components of the hostname in reverse order. For example, the information for **host.example.com** is stored in **com./example./host**. For top-level domains like **com** this can create a large number of subdirectories which on some filesystems can exhaust some limits. Moreover, the performance of lookups in directory with thousands of entries can be fairly slow depending on the filesystem implementation.

Additionally enabling **SingleThreadDelivery** has the added effect of single-threading mail delivery to a destination. This can be quite helpful if the remote machine is running an SMTP server that is easily overloaded or cannot accept more than a single connection at a time, but can cause some messages to be punted to a future queue run. It also applies to *all* hosts, so setting this because you have one machine on site that runs some software that is easily overrun can cause mail to other hosts to be slowed down. If this option is set, you probably want to set the **MinQueueAge** option as well and run the queue fairly frequently; this way jobs that are skipped because another *sendmail* is talking to the same host will be tried again quickly rather than being delayed for a long time.

The disk based host information is stored in a subdirectory of the **mqueue** directory called **.hoststat**[7]. Removing this directory and its subdirectories has an effect similar to the *purgestat* command and is completely safe. However, *purgestat* only removes expired (Timeout.hoststatus) data. The information in these directories can be perused with the *hoststat* command, which will indicate the host name, the last access, and the status of that access. An asterisk in the left most column indicates that a *sendmail* process currently has the host locked for mail delivery.

The disk based connection information is treated the same way as memory based connection information for the purpose of timeouts. By default, information about host failures is valid for 30 minutes. This can be adjusted with the **Timeout.hoststatus** option.

The connection information stored on disk may be expired at any time with the *purgestat* command or by invoking sendmail with the **−bH** switch. The connection information may be viewed with the *hoststat* command or by invoking sendmail with the **−bh** switch.

### 2.5. The Service Switch

The implementation of certain system services such as host and user name lookup is controlled by the service switch. If the host operating system supports such a switch, and sendmail knows about it, *sendmail* will use the native version. Ultrix, Solaris, and DEC OSF/1 are examples of such systems[8].

If the underlying operating system does not support a service switch (e.g., SunOS 4.X, HP-UX, BSD) then *sendmail* will provide a stub implementation. The **ServiceSwitchFile** option points to the name of a file that has the service definitions. Each line has the name of a service and the possible implementations of that service. For example, the file:

```
hosts     dns files nis
aliases   files nis
```

will ask *sendmail* to look for hosts in the Domain Name System first. If the requested host name is not found, it tries local files, and if that fails it tries NIS. Similarly, when looking for aliases it will try the local files first followed by NIS.

Notice: since *sendmail* must access MX records for correct operation, it will use DNS if it is configured in the **ServiceSwitchFile** file. Hence an entry like

```
hosts     files dns
```

will not avoid DNS lookups even if a host can be found in /etc/hosts.

Service switches are not completely integrated. For example, despite the fact that the host entry listed in the above example specifies to look in NIS, on SunOS this won't happen because the system implementation of *gethostbyname* (3) doesn't understand this.

### 2.6. The Alias Database

After recipient addresses are read from the SMTP connection or command line they are parsed by ruleset 0, which must resolve to a {*mailer*, *host*, *address*} triple. If the flags selected by the *mailer* include the **A** (aliasable) flag, the *address* part of the triple is looked up as the key (i.e.,

---

[7]This is the usual value of the **HostStatusDirectory** option; it can, of course, go anywhere you like in your filesystem.

[8]HP-UX 10 has service switch support, but since the APIs are apparently not available in the libraries *sendmail* does not use the native service switch in this release.

the left hand side) in the alias database. If there is a match, the address is deleted from the send queue and all addresses on the right hand side of the alias are added in place of the alias that was found. This is a recursive operation, so aliases found in the right hand side of the alias are similarly expanded.

The alias database exists in two forms. One is a text form, maintained in the file */etc/mail/aliases.* The aliases are of the form

    name: name1, name2, ...

Only local names may be aliased; e.g.,

    eric@prep.ai.MIT.EDU: eric@CS.Berkeley.EDU

will not have the desired effect (except on prep.ai.MIT.EDU, and they probably don't want me)[9]. Aliases may be continued by starting any continuation lines with a space or a tab or by putting a backslash directly before the newline. Blank lines and lines beginning with a sharp sign ("#") are comments.

The second form is processed by the *ndbm* (3)[10] or the Berkeley DB library. This form is in the file */etc/mail/aliases.db* (if using NEWDB) or */etc/mail/aliases.dir* and */etc/mail/aliases.pag* (if using NDBM). This is the form that *sendmail* actually uses to resolve aliases. This technique is used to improve performance.

The control of search order is actually set by the service switch. Essentially, the entry

    O AliasFile=switch:aliases

is always added as the first alias entry; also, the first alias file name without a class (e.g., without "nis:" on the front) will be used as the name of the file for a ''files'' entry in the aliases switch. For example, if the configuration file contains

    O AliasFile=/etc/mail/aliases

and the service switch contains

    aliases    nis files nisplus

then aliases will first be searched in the NIS database, then in /etc/mail/aliases, then in the NIS+ database.

You can also use NIS-based alias files. For example, the specification:

    O AliasFile=/etc/mail/aliases
    O AliasFile=nis:mail.aliases@my.nis.domain

will first search the /etc/mail/aliases file and then the map named "mail.aliases" in "my.nis.domain". Warning: if you build your own NIS-based alias files, be sure to provide the **−l** flag to *makedbm*(8) to map upper case letters in the keys to lower case; otherwise, aliases with upper case letters in their names won't match incoming addresses.

Additional flags can be added after the colon exactly like a **K** line — for example:

    O AliasFile=nis:−N mail.aliases@my.nis.domain

will search the appropriate NIS map and always include null bytes in the key. Also:

    O AliasFile=nis:−f mail.aliases@my.nis.domain

will prevent sendmail from downcasing the key before the alias lookup.

---

[9]Actually, any mailer that has the 'A' mailer flag set will permit aliasing; this is normally limited to the local mailer.

[10]The *gdbm* package does not work.

### 2.6.1. Rebuilding the alias database

The *hash* or *dbm* version of the database may be rebuilt explicitly by executing the command

newaliases

This is equivalent to giving *sendmail* the **−bi** flag:

/usr/sbin/sendmail −bi

If you have multiple aliases databases specified, the **−bi** flag rebuilds all the database types it understands (for example, it can rebuild NDBM databases but not NIS databases).

### 2.6.2. Potential problems

There are a number of problems that can occur with the alias database. They all result from a *sendmail* process accessing the DBM version while it is only partially built. This can happen under two circumstances: 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.

Sendmail has three techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, it locks the database source file during the rebuild — but that may not work over NFS or if the file is unwritable. Third, at the end of the rebuild it adds an alias of the form

@: @

(which is not normally legal). Before *sendmail* will access the database, it checks to insure that this entry exists[11].

### 2.6.3. List owners

If an error occurs on sending to a certain address, say "*x*", *sendmail* will look 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:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
        sam@matisse
owner-unix-wizards: unix-wizards-request
unix-wizards-request: eric@ucbarpa
```

would cause "eric@ucbarpa" to get the error that will occur when someone sends to unix-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, the "owner-" address normally points at the "-request" address; this causes messages to go out with the typical Internet convention of using ''*list*-request'' as the return address.

## 2.7. User Information Database

This option is deprecated, use virtusertable and genericstable instead as explained in *cf/README*. If you have a version of *sendmail* with the user information database compiled in, and

[11]The **AliasWait** option is required in the configuration for this action to occur. This should normally be specified.

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.

### 2.8. Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may 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. Note that aliases are fully expanded before forward files are referenced. For example, if the home directory for user "mckusick" has a .forward file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for "mckusick" will be 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 generally using the **ForwardPath** option. If you change this, you will have to inform your user base of the change; .forward is pretty well incorporated into the collective subconscious.

### 2.9. Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into *sendmail* that cannot be changed without changing the code. These built-ins are described here.

#### 2.9.1. Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses. This is intended for mailing lists.

The Errors-To: header was created in the bad old days when UUCP didn't understand the distinction between an envelope and a header; this was a hack to provide what should now be passed as the envelope sender address. It should go away. It is only used if the **UseErrorsTo** option is set.

The Errors-To: header is officially deprecated and will go away in a future release.

#### 2.9.2. Apparently-To:

RFC 822 requires at least one recipient field (To:, Cc:, or Bcc: line) in every message. If a message comes in with no recipients listed in the message then *sendmail* will adjust the header based on the "NoRecipientAction" option. One of the possible actions is to add an "Apparently-To:" header line for any recipients it is aware of.

The Apparently-To: header is non-standard and is both deprecated and strongly discouraged.

#### 2.9.3. Precedence

The Precedence: header can be used as a crude control of message priority. It tweaks the sort order in the queue and can be configured to change the message timeout values. The precedence of a message also controls how delivery status notifications (DSNs) are processed for that message.

### 2.10. IDENT Protocol Support

*Sendmail* supports the IDENT protocol as defined in RFC 1413. Note that the RFC states a client should wait at least 30 seconds for a response. The default Timeout.ident is 5 seconds as many sites have adopted the practice of dropping IDENT queries. This has lead to delays processing mail. Although this enhances identification of the author of an email message by doing a "call back'' to the originating system to include the owner of a particular TCP connection in the audit trail it is in no sense perfect; a determined forger can easily spoof the IDENT protocol. The following description is excerpted from RFC 1413:

> 6. Security Considerations
>
> The information returned by this protocol is at most as trustworthy as the host providing it OR the organization operating the host. For example, a PC in an open lab has few if any controls on it to prevent a user from having this protocol return any identifier the user wants. Likewise, if the host has been compromised the information returned may be completely erroneous and misleading.
>
> The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. At worst, it can provide misleading, incorrect, or maliciously incorrect information.
>
> The use of the information returned by this protocol for other than auditing is strongly discouraged. Specifically, using Identification Protocol information to make access control decisions - either as the primary method (i.e., no other checks) or as an adjunct to other methods may result in a weakening of normal host security.
>
> An Identification server may reveal information about users, entities, objects or processes which might normally be considered private. An Identification server provides service which is a rough analog of the CallerID services provided by some phone companies and many of the same privacy considerations and arguments that apply to the CallerID service apply to Identification. If you wouldn't run a "finger" server due to privacy considerations you may not want to run this protocol.

In some cases your system may not work properly with IDENT support due to a bug in the TCP/IP implementation. The symptoms will be that for some hosts the SMTP connection will be closed almost immediately. If this is true or if you do not want to use IDENT, you should set the IDENT timeout to zero; this will disable the IDENT protocol.

## 3. ARGUMENTS

The complete list of arguments to *sendmail* is described in detail in Appendix A. Some important arguments are described here.

### 3.1. Queue Interval

The amount of time between forking a process to run through the queue is defined by the **−q** flag. If you run with delivery mode set to **i** or **b** this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in **q** mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue. (See also the MinQueueAge option.)

RFC 1123 section 5.3.1.1 says that this value should be at least 30 minutes (although that probably doesn't make sense if you use ``queue-only'' mode).

Notice: the meaning of the interval time depends on whether normal queue runners or persistent queue runners are used. For the former, it is the time between subsequent starts of a queue run. For the latter, it is the time sendmail waits after a persistent queue runner has finished its work to start the next one. Hence for persistent queue runners this interval should be very low, typically no more than two minutes.

### 3.2. Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your */etc/rc* file using the **−bd** flag. The **−bd** flag and the **−q** flag may be combined in one call:

/usr/sbin/sendmail −bd −q30m

An alternative approach is to invoke sendmail from *inetd*(8) (use the **−bs** −Am flags to ask sendmail to speak SMTP on its standard input and output and to run as MTA). This works and allows you to wrap *sendmail* in a TCP wrapper program, but may be a bit slower since the configuration file has to be re-read on every message that comes in. If you do this, you still need to have a *sendmail* running to flush the queue:

/usr/sbin/sendmail −q30m

### 3.3. Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the **−q** flag (with no value). It is entertaining to use the **−v** flag (verbose) when this is done to watch what happens:

/usr/sbin/sendmail −q −v

You can also limit the jobs to those with a particular queue identifier, recipient, sender, quarantine reason, or queue group 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, "−qSstring" limits the run to particular senders, "−qIstring" limits it to particular queue identifiers, and "−qQstring" limits it to particular quarantined reasons and only operated on quarantined queue items, and "−qGstring" limits it to a particular queue group. The named queue group will be run even if it is set to have 0 runners. You may also place an **!** before the **I** or **R** or **S** or **Q** to indicate that jobs are limited to not including a particular queue identifier, recipient or sender. For example, "−q!Rseattle" limits the queue run to jobs that do not have the string "seattle" somewhere in one of the recipient addresses. Should you need to terminate the queue jobs currently active then a SIGTERM to the parent of the process (or processes) will cleanly stop the jobs.

### 3.4. Debugging

There are a fairly large number of debug flags built into *sendmail*. Each debug flag has a category and a level. Higher levels increase the level of debugging activity; in most cases, this means to print out more information. The convention is that levels greater than nine are "absurd," i.e., they print out so much information that you wouldn't normally want to see them except for debugging that particular piece of code.

You should **never** run a production sendmail server in debug mode. Many of the debug flags will result in debug output being sent over the SMTP channel unless the option **−D** is used. This will confuse many mail programs. However, for testing purposes, it can be useful when sending mail manually via telnet to the port you are using while debugging.

A debug category is either an integer, like 42, or a name, like ANSI. You can specify a range of numeric debug categories using the syntax 17-42. You can specify a set of named debug categories using a glob pattern like "sm_trace_*". At present, only "*" and "?" are supported in these glob patterns.

Debug flags are set using the **−d** option; the syntax is:

| | |
|---|---|
| debug-flag: | **−d** debug-list |
| debug-list: | debug-option [ , debug-option ]* |
| debug-option: | debug-categories [ . debug-level ] |
| debug-categories: | integer \| integer − integer \| category-pattern |
| category-pattern: | [a-zA-Z_*?][a-zA-Z0-9_*?]* |
| debug-level: | integer |

where spaces are for reading ease only. For example,

| | |
|---|---|
| −d12 | Set category 12 to level 1 |
| −d12.3 | Set category 12 to level 3 |
| −d3−17 | Set categories 3 through 17 to level 1 |
| −d3−17.4 | Set categories 3 through 17 to level 4 |
| −dANSI | Set category ANSI to level 1 |
| −dsm_trace_*.3 | Set all named categories matching sm_trace_* to level 3 |

For a complete list of the available debug flags you will have to look at the code and the *TRACEFLAGS* file in the sendmail distribution (they are too dynamic to keep this document up to date). For a list of named debug categories in the sendmail binary, use

```
ident /usr/sbin/sendmail | grep Debug
```

### 3.5. Changing the Values of Options

Options can be overridden using the **−o** or **−O** command line flags. For example,

```
/usr/sbin/sendmail −oT2m
```

sets the **T** (timeout) option to two minutes for this run only; the equivalent line using the long option name is

```
/usr/sbin/sendmail -OTimeout.queuereturn=2m
```

Some options have security implications. Sendmail allows you to set these, but relinquishes its set-user-ID or set-group-ID permissions thereafter[12].

### 3.6. Trying a Different Configuration File

An alternative configuration file can be specified using the **−C** flag; for example,

```
/usr/sbin/sendmail −Ctest.cf −oQ/tmp/mqueue
```

uses the configuration file *test.cf* instead of the default */etc/mail/sendmail.cf.* If the **−C** flag has no value it defaults to *sendmail.cf* in the current directory.

*Sendmail* gives up set-user-ID root permissions (if it has been installed set-user-ID root) when you use this flag, so it is common to use a publicly writable directory (such as /tmp) as the queue directory (QueueDirectory or Q option) while testing.

### 3.7. 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,

```
/usr/sbin/sendmail −X /tmp/traffic −bd
```

will log all traffic in the file */tmp/traffic*.

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.

---

[12]That is, it sets its effective uid to the real uid; thus, if you are executing as root, as from root's crontab file or during system startup the root permissions will still be honored.

### 3.8. Testing Configuration Files

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

sendmail −bt −Ctest.cf

which would read the configuration file "test.cf" and enter test mode. In this mode, you enter lines of the form:

rwset address

where *rwset* is the rewriting set you want to use and *address* is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. 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

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the "−d21" flag to turn on more debugging. For example,

sendmail −bt −d21.99

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

You should be warned that internally, *sendmail* applies ruleset 3 to all addresses. In test mode 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

As of version 8.7, some other syntaxes are available in test mode:

| | |
|---|---|
| .D x value | defines macro *x* to have the indicated *value*. This is useful when debugging rules that use the **$&***x* syntax. |
| .C c value | adds the indicated *value* to class *c*. |
| =S ruleset | dumps the contents of the indicated ruleset. |
| −d debug-spec | is equivalent to the command-line flag. |

Version 8.9 introduced more features:

| | |
|---|---|
| ? | shows a help message. |
| =M | display the known mailers. |
| $m | print the value of macro m. |
| $=c | print the contents of class c. |
| /mx host | returns the MX records for 'host'. |
| /parse address | parse address, returning the value of *crackaddr*, and the parsed address. |
| /try mailer addr | rewrite address into the form it will have when presented to the indicated mailer. |
| /tryflags flags | set flags used by parsing. The flags can be 'H' for Header or 'E' for Envelope, and 'S' for Sender or 'R' for Recipient. These can be combined, 'HR' sets flags for header recipients. |

/canon hostname try to canonify hostname.

/map mapname key
look up 'key' in the indicated 'mapname'.

/quit quit address test mode.

### 3.9. Persistent Host Status Information

When **HostStatusDirectory** is enabled, information about the status of hosts is maintained on disk and can thus be shared between different instantiations of *sendmail*. The status of the last connection with each remote host may be viewed with the command:

sendmail −bh

This information may be flushed with the command:

sendmail −bH

Flushing the information prevents new *sendmail* processes from loading it, but does not prevent existing processes from using the status information that they already have.

## 4. TUNING

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. For example, the line "O Timeout.queuereturn=5d" sets option "Timeout.queuereturn" to the value "5d" (five days).

Most of these options have appropriate defaults for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for their mail 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.

All versions of *sendmail* prior to 8.7 had single character option names. As of 8.7, options have long (multi-character names). Although old short names are still accepted, most new options do not have short equivalents.

This section only describes the options you are most likely to want to tweak; read section 5 for more details.

### 4.1. Timeouts

All time intervals are set using a scaled syntax. For example, "10m" represents ten minutes, whereas "2h30m" represents two and a half hours. The full set of scales is:

| | |
|---|---|
| s | seconds |
| m | minutes |
| h | hours |
| d | days |
| w | weeks |

#### 4.1.1. Queue interval

The argument to the **−q** flag specifies how often a sub-daemon will run the queue. This is typically set to between fifteen minutes and one hour. If not set, or set to zero, the queue will not be run automatically. RFC 1123 section 5.3.1.1 recommends that this be at least 30 minutes. Should you need to terminate the queue jobs currently active then a SIGTERM to the parent of the process (or processes) will cleanly stop the jobs.

#### 4.1.2. Read timeouts

Timeouts all have option names "Timeout.*suboption*". Most of these control SMTP operations. The recognized *suboption*s, their default values, and the minimum values allowed by

RFC 2821 section 4.5.3.2 (or RFC 1123 section 5.3.2) are:

connect — The time to wait for an SMTP connection to open (the *connect*(2) system call) [0, unspecified]. If zero, uses the kernel default. In no case can this option extend the timeout longer than the kernel provides, but it can shorten it. This is to get around kernels that provide an absurdly long connection timeout (90 minutes in one case).

iconnect — The same as *connect,* except it applies only to the initial attempt to connect to a host for a given message [0, unspecified]. The concept is that this should be very short (a few seconds); hosts that are well connected and responsive will thus be serviced immediately. Hosts that are slow will not hold up other deliveries in the initial delivery attempt.

aconnect — [0, unspecified] The overall timeout waiting for all connection for a single delivery attempt to succeed. If 0, no overall limit is applied. This can be used to restrict the total amount of time trying to connect to a long list of host that could accept an e-mail for the recipient. This timeout does not apply to **FallbackMXhost**, i.e., if the time is exhausted, the **FallbackMXhost** is tried next.

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 a RCPT command [1h, 5m]. This should be long because it could be pointing at a list that takes a long time to expand (see below).

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].

ident‡ — The timeout waiting for a reply to an IDENT query [5s[13], unspecified].

lhlo — The wait for a reply to an LMTP LHLO command [2m, unspecified].

auth — The timeout for a reply in an SMTP AUTH dialogue [10m, unspecified].

starttls — The timeout for a reply to an SMTP STARTTLS command and the TLS handshake [1h, unspecified].

---

[13]On some systems the default is zero to turn the protocol off entirely.

fileopen‡ The timeout for opening .forward and :include: files [60s, none].

control‡ The timeout for a complete control socket transaction to complete [2m, none].

hoststatus‡ How long status information about a host (e.g., host down) will be cached before it is considered stale [30m, unspecified].

resolver.retrans‡ The resolver's retransmission time interval (in seconds) [varies]. Sets both *Timeout.resolver.retrans.first* and *Timeout.resolver.retrans.normal*.

resolver.retrans.first‡
The resolver's retransmission time interval (in seconds) for the first attempt to deliver a message [varies].

resolver.retrans.normal‡
The resolver's retransmission time interval (in seconds) for all resolver lookups except the first delivery attempt [varies].

resolver.retry‡ The number of times to retransmit a resolver query. Sets both *Timeout.resolver.retry.first* and *Timeout.resolver.retry.normal* [varies].

resolver.retry.first‡
The number of times to retransmit a resolver query for the first attempt to deliver a message [varies].

resolver.retry.normal‡
The number of times to retransmit a resolver query for all resolver lookups except the first delivery attempt [varies].

For compatibility with old configuration files, if no *suboption* is specified, all the timeouts marked with a dagger (†) are set to the indicated value. All but those marked with a double dagger (‡) apply to client SMTP.

For example, the lines:

O Timeout.command=25m
O Timeout.datablock=3h

sets the server SMTP command timeout to 25 minutes and the input data block timeout to three hours.

### 4.1.3. Message timeouts

After sitting in the queue for a few days, an undeliverable message will time out. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to five days. It is sometimes considered convenient to also send a warning message if the message is in the queue longer than a few hours (assuming you normally have good connectivity; if your messages normally took several hours to send you wouldn't want to do this because it wouldn't be an unusual event). These timeouts are set using the **Timeout.queuereturn** and **Timeout.queuewarn** options in the configuration file (previously both were set using the **T** option).

If the message is submitted using the NOTIFY SMTP extension, warning messages will only be sent if NOTIFY=DELAY is specified. The queuereturn and queuewarn timeouts can be further qualified with a tag based on the Precedence: field in the message; they must be one of "urgent" (indicating a positive non-zero precedence), "normal" (indicating a zero precedence), or "non-urgent" (indicating negative precedences). For example, setting "Timeout.queuewarn.urgent=1h" sets the warning timeout for urgent messages only to one hour. The default if no precedence is indicated is to set the timeout for all precedences. If the message has a normal (default) precedence and it is a delivery status notification (DSN), **Timeout.queuereturn.dsn** and **Timeout.queuewarn.dsn** can be used to give an alternative warn and return time for DSNs. The value "now" can be used for -O Timeout.queuereturn to return entries immediately during a queue run, e.g., to bounce messages independent of their time in the queue.

Since these options are global, and since you cannot know *a priori* 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 **Timeout.queuewarn** value can be piggybacked on the **T** option by indicating a time after which a warning message should be sent; the two timeouts are separated by a slash. For example, the line

```
OT5d/4h
```

causes email to fail after five days, but a warning message will be sent after four hours. This should be large enough that the message will have been tried several times.

### 4.2. Forking During Queue Runs

By setting the **ForkEachJob** (**Y**) option, *sendmail* will fork before each individual message while running the queue. This option was used with earlier releases to prevent *sendmail* from consuming large amounts of memory. It should no longer be necessary with *sendmail* 8.12. If the **ForkEachJob** option is not set, *sendmail* will keep track of hosts that are down during a queue run, which can improve performance dramatically.

If the **ForkEachJob** option is set, *sendmail* cannot use connection caching.

### 4.3. 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 (which is determined from the Precedence: header) times the “work class factor” and the number of recipients times the “work recipient factor.” The priority is used to order the queue. Higher numbers for the priority mean that the message will be 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. Since the number of recipients affects the amount of load a message presents to the system, this is also included into the priority.

The recipient and class factors can be set in the configuration file using the **RecipientFactor** (**y**) and **ClassFactor** (**z**) options respectively. They default to 30000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

$$pri = msgsize - (class \times \mathbf{ClassFactor}) + (nrcpt \times \mathbf{RecipientFactor})$$

(Remember, higher values for this parameter actually mean that the job will be 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 **RetryFactor** (**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 will tend to fail again in the future. The **RetryFactor** option defaults to 90000.

### 4.4. Load Limiting

*Sendmail* can be asked to queue (but not deliver) mail if the system load average gets too high using the **QueueLA** (**x**) option. When the load average exceeds the value of the **QueueLA** option, the delivery mode is set to **q** (queue only) if the **QueueFactor** (**q**) option divided by the difference in the current load average and the **QueueLA** option plus one is less than the priority of the message

— that is, the message is queued iff:

$$pri > \frac{\textbf{QueueFactor}}{LA - \textbf{QueueLA} + 1}$$

The **QueueFactor** option defaults to 600000, so each point of load average is worth 600000 priority points (as described above).

For drastic cases, the **RefuseLA** (**X**) option defines a load average at which *sendmail* will refuse to accept network connections. Locally generated mail, i.e., mail which is not submitted via SMTP (including incoming UUCP mail), is still accepted. Notice that the MSP submits mail to the MTA via SMTP, and hence mail will be queued in the client queue in such a case. Therefore it is necessary to run the client mail queue periodically.

### 4.5. Resource Limits

*Sendmail* has several parameters to control resource usage. Besides those mentionted in the previous section, there are at least **MaxDaemonChildren**, **ConnectionRateThrottle**, **MaxQueueChildren**, and **MaxRunnersPerQueue**. The latter two limit the number of *sendmail* processes that operate on the queue. These are discussed in the section ''Queue Group Declaration''. The former two can be used to limit the number of incoming connections. Their appropriate values depend on the host operating system and the hardware, e.g., amount of memory. In many situations it might be useful to set limits to prevent to have too many *sendmail* processes, however, these limits can be abused to mount a denial of service attack. For example, if **MaxDaemonChildren=10** then an attacker needs to open only 10 SMTP sessions to the server, leave them idle for most of the time, and no more connections will be accepted. If this option is set then the timeouts used in a SMTP session should be lowered from their default values to their minimum values as specified in RFC 2821 and listed in section 4.1.2.

### 4.6. Measures against Denial of Service Attacks

*Sendmail* has some built-in measures against simple denial of service (DoS) attacks. The SMTP server by default slows down if too many bad commands are issued or if some commands are repeated too often within a session. Details can be found in the source file **sendmail/srvrsmtp.c** by looking for the macro definitions of **MAXBADCOMMANDS**, **MAXNOOPCOMMANDS**, **MAXHELOCOMMANDS**, **MAXVRFYCOMMANDS**, and **MAXETRNCOMMANDS**. If an SMTP command is issued more often than the corresponding **MAXcmdCOMMANDS** value, then the response is delayed exponentially, starting with a sleep time of one second, up to a maximum of four minutes (as defined by **MAXTIMEOUT**). If the option **MaxDaemonChildren** is set to a value greater than zero, then this could make a DoS attack even worse since it keeps a connection open longer than necessary. Therefore a connection is terminated with a 421 SMTP reply code if the number of commands exceeds the limit by a factor of two and **MAXBADCOMMANDS** is set to a value greater than zero (the default is 25).

### 4.7. Delivery Mode

There are a number of delivery modes that *sendmail* can operate in, set by the **DeliveryMode** (**d**) configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

| | |
|---|---|
| i | deliver interactively (synchronously) |
| b | deliver in background (asynchronously) |
| q | queue only (don't deliver) |
| d | defer delivery attempts (don't deliver) |

There are tradeoffs. Mode "i" gives the sender the quickest feedback, but may slow down some mailers and is hardly ever necessary. Mode "b" delivers promptly but can cause large numbers of processes if you have a mailer that takes a long time to deliver a message. Mode "q" minimizes the load on your machine, but means that delivery may be delayed for up to the queue interval. Mode "d" is identical to mode "q" except that it also prevents lookups in maps including the **-D** flag from working during the initial queue phase; it is intended for ''dial on demand'' sites where DNS lookups might cost real money. Some simple error messages (e.g., host unknown during the SMTP protocol) will be delayed using this mode. Mode "b" is the usual default.

If you run in mode "q" (queue only), "d" (defer), or "b" (deliver in background) *sendmail* will not expand aliases and follow .forward files upon initial receipt of the mail. This speeds up the response to RCPT commands. Mode "i" should not be used by the SMTP server.

### 4.8. Log Level

The level of logging can be set for *sendmail*. The default using a standard configuration table is level 9. The levels are as follows:

| | |
|---|---|
| 0 | Minimal logging. |
| 1 | Serious system failures and potential security problems. |
| 2 | Lost communications (network problems) and protocol failures. |
| 3 | Other serious failures, malformed addresses, transient forward/include errors, connection timeouts. |
| 4 | Minor failures, out of date alias databases, connection rejections via check_ rulesets. |
| 5 | Message collection statistics. |
| 6 | Creation of error messages, VRFY and EXPN commands. |
| 7 | Delivery failures (host or user unknown, etc.). |
| 8 | Successful deliveries and alias database rebuilds. |
| 9 | Messages being deferred (due to a host being down, etc.). |
| 10 | Database expansion (alias, forward, and userdb lookups) and authentication information. |
| 11 | NIS errors and end of job processing. |
| 12 | Logs all SMTP connections. |
| 13 | Log bad user shells, files with improper permissions, and other questionable situations. |
| 14 | Logs refused connections. |
| 15 | Log all incoming and outgoing SMTP commands. |
| 20 | Logs attempts to run locked queue files. These are not errors, but can be useful to note if your queue appears to be clogged. |
| 30 | Lost locks (only if using lockf instead of flock). |

Additionally, values above 64 are reserved for extremely verbose debugging output. No normal site would ever set these.

### 4.9. File Modes

The modes used for files depend on what functionality you want and the level of security you require. In many cases *sendmail* does careful checking of the modes of files and directories to avoid accidental compromise; if you want to make it possible to have group-writable support files you may need to use the **DontBlameSendmail** option to turn off some of these checks.

#### 4.9.1. To suid or not to suid?

*Sendmail* is no longer installed set-user-ID to root. sendmail/SECURITY explains how to configure and install *sendmail* without set-user-ID to root but set-group-ID which is the default configuration starting with 8.12.

The daemon usually runs as root, unless other measures are taken. At the point where *sendmail* is about to *exec* (2) a mailer, it checks to see if the userid is zero (root); if so, it resets the userid and groupid to a default (set by the **U=** equate in the mailer line; if that is not set, the **DefaultUser** option is used). This can be overridden by setting the **S** flag to the mailer for mailers that are trusted and must be called as root. However, this will cause mail processing to be accounted (using *sa* (8)) to root rather than to the user sending the mail.

A middle ground is to set the **RunAsUser** option. This causes *sendmail* to become the indicated user as soon as it has done the startup that requires root privileges (primarily, opening the SMTP socket). If you use **RunAsUser**, the queue directory (normally */var/spool/mqueue*) should be owned by that user, and all files and databases (including user *.forward* files, alias files, :include: files, and external databases) must be readable by that user. Also, since sendmail will not be able to change its uid, delivery to programs or files will be marked as unsafe, e.g., undeliverable, in *.forward*, aliases, and :include: files. Administrators can override this by setting the **DontBlameSendmail** option to the setting **NonRootSafeAddr**. **RunAsUser** is probably best suited for firewall configurations that don't have regular user logins. If the option is used on a system which performs local delivery, then the local delivery agent must have the proper permissions (i.e., usually set-user-ID root) since it will be invoked by the **RunAsUser**, not by root.

### 4.9.2. Turning off security checks

*Sendmail* is very particular about the modes of files that it reads or writes. For example, by default it will refuse to read most files that are group writable on the grounds that they might have been tampered with by someone other than the owner; it will even refuse to read files in group writable directories. Also, sendmail will refuse to create a new aliases database in an unsafe directory. You can get around this by manually creating the database file as a trusted user ahead of time and then rebuilding the aliases database with **newaliases**.

If you are *quite* sure that your configuration is safe and you want *sendmail* to avoid these security checks, you can turn off certain checks using the **DontBlameSendmail** option. This option takes one or more names that disable checks. In the descriptions that follow, "unsafe directory" means a directory that is writable by anyone other than the owner. The values are:

Safe No special handling.

AssumeSafeChown
: Assume that the *chown* system call is restricted to root. Since some versions of UNIX permit regular users to give away their files to other users on some filesystems, *sendmail* often cannot assume that a given file was created by the owner, particularly when it is in a writable directory. You can set this flag if you know that file giveaway is restricted on your system.

ClassFileInUnsafeDirPath
: When reading class files (using the **F** line in the configuration file), allow files that are in unsafe directories.

DontWarnForwardFileInUnsafeDirPath
: Prevent logging of unsafe directory path warnings for non-existent forward files.

ErrorHeaderInUnsafeDirPath
: Allow the file named in the **ErrorHeader** option to be in an unsafe directory.

FileDeliveryToHardLink
: Allow delivery to files that are hard links.

FileDeliveryToSymLink
: Allow delivery to files that are symbolic links.

ForwardFileInGroupWritableDirPath
: Allow *.forward* files in group writable directories.

ForwardFileInUnsafeDirPath
: Allow *.forward* files in unsafe directories.

ForwardFileInUnsafeDirPathSafe
: Allow a *.forward* file that is in an unsafe directory to include references to program and files.

GroupReadableKeyFile
: Accept a group-readable key file for STARTTLS.

GroupReadableSASLDBFile
: Accept a group-readable Cyrus SASL password file.

GroupWritableAliasFile
: Allow group-writable alias files.

GroupWritableDirPathSafe
: Change the definition of "unsafe directory" to consider group-writable directories to be safe. World-writable directories are always unsafe.

GroupWritableForwardFile
: Allow group writable *.forward* files.

GroupWritableForwardFileSafe
: Accept group-writable *.forward* files as safe for program and file delivery.

GroupWritableIncludeFile
: Allow group wriable *:include:* files.

GroupWritableIncludeFileSafe
: Accept group-writable *:include:* files as safe for program and file delivery.

GroupWritableSASLDBFile
: Accept a group-writable Cyrus SASL password file.

HelpFileInUnsafeDirPath
: Allow the file named in the **HelpFile** option to be in an unsafe directory.

IncludeFileInGroupWritableDirPath
: Allow *:include:* files in group writable directories.

IncludeFileInUnsafeDirPath
: Allow *:include:* files in unsafe directories.

IncludeFileInUnsafeDirPathSafe
: Allow a *:include:* file that is in an unsafe directory to include references to program and files.

InsufficientEntropy
: Try to use STARTTLS even if the PRNG for OpenSSL is not properly seeded despite the security problems.

LinkedAliasFileInWritableDir
: Allow an alias file that is a link in a writable directory.

LinkedClassFileInWritableDir
: Allow class files that are links in writable directories.

LinkedForwardFileInWritableDir
: Allow *.forward* files that are links in writable directories.

LinkedIncludeFileInWritableDir
: Allow *:include:* files that are links in writable directories.

LinkedMapInWritableDir
: Allow map files that are links in writable directories. This includes alias database files.

LinkedServiceSwitchFileInWritableDir
: Allow the service switch file to be a link even if the directory is writable.

MapInUnsafeDirPath
: Allow maps (e.g., *hash*, *btree*, and *dbm* files) in unsafe directories. This includes alias database files.

NonRootSafeAddr
: Do not mark file and program deliveries as unsafe if sendmail is not running with root privileges.

RunProgramInUnsafeDirPath
: Run programs that are in writable directories without logging a warning.

RunWritableProgram
: Run programs that are group- or world-writable without logging a warning.

TrustStickyBit
: Allow group or world writable directories if the sticky bit is set on the directory. Do not set this on systems which do not honor the sticky bit on directories.

WorldWritableAliasFile
: Accept world-writable alias files.

WorldWritableForwardfile
: Allow world writable *.forward* files.

WorldWritableIncludefile
: Allow world wriable *:include:* files.

WriteMapToHardLink
: Allow writes to maps that are hard links.

WriteMapToSymLink
: Allow writes to maps that are symbolic links.

WriteStatsToHardLink
: Allow the status file to be a hard link.

WriteStatsToSymLink
: Allow the status file to be a symbolic link.

### 4.10. Connection Caching

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 and LPC 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 RSET command. It is not an error if this fails; instead, the connection is closed and reopened.

Two parameters control the connection cache. The **ConnectionCacheSize** (**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. Never set this higher than 4.

The **ConnectionCacheTimeout** (**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.

### 4.11. Name Server Access

Control of host address lookups is set by the **hosts** service entry in your service switch file. If you are on a system that has built-in service switch support (e.g., Ultrix, Solaris, or DEC OSF/1) then your system is probably configured properly already. Otherwise, *sendmail* will consult the file **/etc/mail/service.switch**, which should be created. *Sendmail* only uses two entries: **hosts** and **aliases**, although system routines may use other services (notably the **passwd** service for user name lookups by *getpwname*).

However, some systems (such as SunOS 4.X) will do DNS lookups regardless of the setting of the service switch entry. In particular, the system routine *gethostbyname*(3) is used to look up host names, and many vendor versions try some combination of DNS, NIS, and file lookup in /etc/hosts without consulting a service switch. *Sendmail* makes no attempt to work around this problem, and the DNS lookup will be done anyway. If you do not have a nameserver 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. If the **hosts** switch entry has the service "dns" listed somewhere in the list, *sendmail* will interpret this to mean a temporary failure and will queue the mail for later processing; otherwise, it ignores the name server data.

The same technique is used to decide whether to do MX lookups. If you want MX support, you *must* have "dns" listed as a service in the **hosts** switch entry.

The **ResolverOptions** (**I**) option allows you to tweak name server options. The command line takes a series of flags as documented in *resolver*(3) (with the leading "RES_" deleted). Each can be preceded by an optional '+' or '−'. For example, the line

```
O ResolverOptions=+AAONLY −DNSRCH
```

turns on the AAONLY (accept authoritative answers only) and turns off the DNSRCH (search the domain path) options. Most resolver libraries default DNSRCH, DEFNAMES, and RECURSE flags on and all others off. If NETINET6 is enabled, most libraries default to USE_INET6 as well. You can also include "HasWildcardMX" to specify that there is a wildcard MX record matching your domain; this turns off MX matching when canonifying names, which can lead to inappropriate canonifications. Use "WorkAroundBrokenAAAA" when faced with a broken nameserver that returns SERVFAIL (a temporary failure) on T_AAAA (IPv6) lookups during hostname canonification. Notice: it might be necessary to apply the same (or similar) options to *submit.cf* too.

Version level 1 configurations (see the section about ‘‘Configuration Version Level’’) 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 don't want to do automatic name extension, don't call $[ ... $].

The search rules for $[ ... $] are somewhat different than usual. If the name being looked up 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, since 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.

To completely turn off all name server access on systems without service switch support (such as SunOS 4.X) you will have to recompile with −DNAMED_BIND=0 and remove −lresolv from the list of libraries to be searched when linking.

### 4.12. 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 from a central mail server 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 automounter.

The **ForwardPath** (**J**) option allows you to set a path of forward files. For example, the config file line

    O ForwardPath=/var/forward/$u:$z/.forward.$w

would first look for a file with the same name as the user's login in /var/forward; if that is not found (or is inaccessible) the file ''.forward.*machinename*'' in the user's home directory is searched. A truly perverse site could also search by sender by using $r, $s, or $f.

If you create a directory such as /var/forward, it should be mode 1777 (that is, the sticky bit should be set). Users should create the files mode 0644. Note that you must use the ForwardFileInUnsafeDirPath and ForwardFileInUnsafeDirPathSafe flags with the **DontBlameSendmail** option to allow forward files in a world writable directory. This might also be used as a denial of service attack (users could create forward files for other users); a better approach might be to create /var/forward mode 0755 and create empty files for each user, owned by that user, mode 0644. If you do this, you don't have to set the DontBlameSendmail options indicated above.

### 4.13. Free Space

On systems that have one of the system calls in the *statfs*(2) family (including *statvfs* and *ustat*), you can specify a minimum number of free blocks on the queue filesystem using the **MinFreeBlocks** (**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.

### 4.14. Maximum Message Size

To avoid overflowing your system with a large message, the **MaxMessageSize** option can be set to set an absolute limit on the size of any one message. This will be advertised in the ESMTP dialogue and checked during message collection.

### 4.15. Privacy Flags

The **PrivacyOptions** (**p**) option allows you to set certain ''privacy'' flags. Actually, many of them don't give you any extra privacy, rather just insisting that client SMTP servers use the HELO command before using certain commands or adding extra headers to indicate possible spoof attempts.

The option takes a series of flag names; the final privacy is the inclusive or of those flags. For example:

    O PrivacyOptions=needmailhelo, noexpn

insists that the HELO or EHLO command be used before a MAIL command is accepted and disables the EXPN command.

The flags are detailed in section 5.6.

### 4.16. Send to Me Too

Beginning with version 8.10, *sendmail* includes by default the (envelope) sender in any list expansions. For example, if "matt" sends to a list that contains "matt" as one of the members he will get a copy of the message. If the **MeToo** option is set to FALSE (in the configuration file or via the command line), this behavior is changed, i.e., the (envelope) sender is excluded in list expansions.

## 5. THE WHOLE SCOOP ON THE CONFIGURATION FILE

This section describes the configuration file in detail.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time *sendmail* starts up, rather than easy for a human to read or write. The configuration file should be generated via the method described in **cf/README**, it should not be edited directly unless someone is familiar with the internals of the syntax described here and it is not possible to achieve the desired result via the default method.

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

### 5.1. R and S — Rewriting Rules

The core of address parsing are the rewriting rules. These are an ordered production system. *Sendmail* scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the 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 are:

**S***n*

Sets the current ruleset being collected to *n*. If you begin a ruleset more than once it appends to the old definition.

**R***lhs rhs comments*

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

Macro expansions of the form **$***x* are performed when the configuration file is read. A literal **$** can be included using **$$**. 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.

#### 5.1.1. The left hand side

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, where *n* is the index in the LHS. For example, if the LHS:

$−:$+

is applied to the input:

UCBARPA:eric

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

| | |
|---|---|
| $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.

### 5.1.2. The right hand side

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:

| | |
|---|---|
| **$***n* | Substitute indefinite token *n* from LHS |
| **$[***name***$]** | Canonicalize *name* |
| **$(***map key* **$@***arguments* **$:***default* **$)** | Generalized keyed mapping function |
| **$>***n* | "Call" ruleset *n* |
| **$#***mailer* | Resolve to *mailer* |
| **$@***host* | Specify *host* |
| **$:***user* | Specify *user* |

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

A host name enclosed between **$[** and **$]** is looked up in the host database(s) and replaced by the canonical name[14]. For example, "$[ftp$]" might become "ftp.CS.Berkeley.EDU" and "$[[128.32.130.2]$]" would become "vangogh.CS.Berkeley.EDU." *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 *arguments* are passed to the map for possible use.

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 expands everything after the ruleset name to the end of the replacement string and then passes that as the initial input to the ruleset. Recursive calls are allowed. For example,

$>0 $>3 $1

expands $1, passes that to ruleset 3, and then passes the result of ruleset 3 to ruleset 0.

The **$#** syntax should *only* be used in ruleset zero, a subroutine of ruleset zero, or rulesets that return decisions (e.g., check_rcpt). It causes evaluation of the ruleset to terminate immediately, and signals to *sendmail* that the address has completely resolved. The complete syntax for ruleset 0 is:

**$#***mailer* **$@***host* **$:***user*

This specifies the {mailer, host, user} 3-tuple necessary to direct the mailer. Note: the third

[14] This is actually completely equivalent to $(host *hostname*$). In particular, a **$:** default can be used.

element ( *user* ) is often also called *address* part. If the mailer is local the host part may be omitted[15]. The *mailer* must be a single word, but the *host* and *user* may 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 **$u** macro. As a special case, if the mailer specified has the **F=@** flag specified and the first character of the **$:** 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 may 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 be used to avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes may precede a **$>** spec; for example:

```
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.

### 5.1.3. Semantics of rewriting rule sets

There are six rewriting sets that have specific semantics. Five of these are related as depicted by figure 1.

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

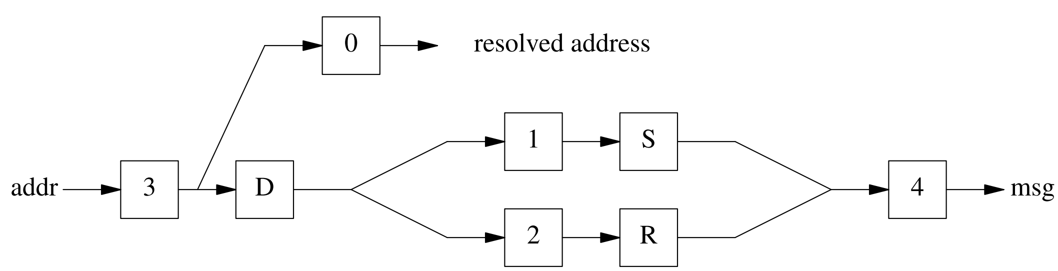


Figure 1 — Rewriting set semantics
D — sender domain addition
S — mailer-specific sender rewriting
R — mailer-specific recipient rewriting

[15] You may want to use it for special "per user" extensions. For example, in the address "jgm+foo@CMU.EDU"; the "+foo" part is not part of the user name, and is passed to the local mailer for local use.

local-part@host-domain-spec

Ruleset three is applied by *sendmail* before doing anything with any address.

If no "@" sign is specified, then the host-domain-spec *may* be appended (box "D" in Figure 1) from the sender address (if the **C** flag is set in the mailer definition corresponding to the *sending* mailer).

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a *{mailer, host, address}* 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. Notice: since the envelope sender address will be used if a delivery status notification must be send, i.e., is may specify a recipient, it is also run through ruleset zero. If ruleset zero returns a temporary error **4xy** then delivery is deferred. This can be used to temporarily disable delivery, e.g., based on the time of the day or other varying parameters. It should not be used to quarantine e-mails.

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.

In addition, ruleset 5 is applied to all local addresses (specifically, those that resolve to a mailer with the 'F=5' flag set) that do not have aliases. This allows a last minute hook for local names.

### 5.1.4. Ruleset hooks

A few extra rulesets are defined as "hooks" that can be defined to get special features. They are all named rulesets. The "check_*" forms all give accept/reject status; falling off the end or returning normally is an accept, and resolving to **$#error** is a reject or quarantine. Quarantining is chosen by specifying **quarantine** in the second part of the mailer triplet:

$#error $@ quarantine $: Reason for quarantine

Many of these can also resolve to the special mailer name **$#discard**; this accepts the message as though it were successful but then discards it without delivery. Note, this mailer cannot be chosen as a mailer in ruleset 0. Note also that all "check_*" rulesets have to deal with temporary failures, especially for map lookups, themselves, i.e., they should return a temporary error code or at least they should make a proper decision in those cases.

#### 5.1.4.1. check_relay

The *check_relay* ruleset is called after a connection is accepted by the daemon. It is not called when sendmail is started using the **−bs** option. It is passed

client.host.name $| client.host.address

where **$|** is a metacharacter separating the two parts. This ruleset can reject connections from various locations. Note that it only checks the connecting SMTP client IP address and hostname. It does not check for third party message relaying. The *check_rcpt* ruleset discussed below usually does third party message relay checking.

#### 5.1.4.2. check_mail

The *check_mail* ruleset is passed the user name parameter of the SMTP MAIL command. It can accept or reject the address.

#### 5.1.4.3. check_rcpt

The *check_rcpt* ruleset is passed the user name parameter of the SMTP RCPT command. It can accept or reject the address.

#### 5.1.4.4. check_data

The *check_data* ruleset is called after the SMTP DATA command, its parameter is the number of recipients. It can accept or reject the command.

#### 5.1.4.5. check_compat

The *check_compat* ruleset is passed

sender-address $| recipient-address

where **$|** is a metacharacter separating the addresses. It can accept or reject mail transfer between these two addresses much like the *checkcompat()* function. Note: while other *check_** rulesets are invoked during the SMTP mail receiption stage (i.e., in the SMTP server), *check_compat* is invoked during the mail delivery stage.

#### 5.1.4.6. check_eoh

The *check_eoh* ruleset is passed

number-of-headers $| size-of-headers

where **$|** is a metacharacter separating the numbers. These numbers can be used for size comparisons with the **arith** map. The ruleset is triggered after all of the headers have been read. It can be used to correlate information gathered from those headers using the **macro** storage map. One possible use is to check for a missing header. For example:

```
Kstorage macro
HMessage-Id: $>CheckMessageId

SCheckMessageId
# Record the presence of the header
R$*                     $: $(storage {MessageIdCheck} $@ OK $) $1
R< $+ @ $+ >            $@ OK
R$*                     $#error $: 553 Header Error

Scheck_eoh
# Check the macro
R$*                     $: < $&{MessageIdCheck} >
# Clear the macro for the next message
R$*                     $: $(storage {MessageIdCheck} $) $1
# Has a Message-Id: header
R< $+ >                 $@ OK
# Allow missing Message-Id: from local mail
R$*                     $: < $&{client_name} >
R< >                    $@ OK
R< $=w >                $@ OK
# Otherwise, reject the mail
R$*                     $#error $: 553 Header Error
```

Keep in mind the Message-Id: header is not a required header and is not a guaranteed spam indicator. This ruleset is an example and should probably not be used in production.

#### 5.1.4.7. check_eom

The *check_eom* ruleset is called after the end of a message, its parameter is the message size. It can accept or reject the message.

#### 5.1.4.8. check_etrn

The *check_etrn* ruleset is passed the parameter of the SMTP ETRN command. It can accept or reject the command.

#### 5.1.4.9. check_expn

The *check_expn* ruleset is passed the user name parameter of the SMTP EXPN command. It can accept or reject the address.

#### 5.1.4.10. check_vrfy

The *check_vrfy* ruleset is passed the user name parameter of the SMTP VRFY command. It can accept or reject the command.

#### 5.1.4.11. trust_auth

The *trust_auth* ruleset is passed the AUTH= parameter of the SMTP MAIL command. It is used to determine whether this value should be trusted. In order to make this decision, the ruleset may make use of the various **${auth_*}** macros. If the ruleset does resolve to the "error" mailer the AUTH= parameter is not trusted and hence not passed on to the next relay.

#### 5.1.4.12. tls_client

The *tls_client* ruleset is called when sendmail acts as server, after a STARTTLS command has been issued, and from *check_mail.* The parameter is the value of **${verify}** and STARTTLS or MAIL, respectively. If the ruleset does resolve to the "error" mailer, the appropriate error code is returned to the client.

#### 5.1.4.13. tls_server

The *tls_server* ruleset is called when sendmail acts as client after a STARTTLS command (should) have been issued. The parameter is the value of **${verify}**. If the ruleset does resolve to the "error" mailer, the connection is aborted (treated as non-deliverable with a permanent or temporary error).

#### 5.1.4.14. tls_rcpt

The *tls_rcpt* ruleset is called each time before a RCPT TO command is sent. The parameter is the current recipient. If the ruleset does resolve to the "error" mailer, the RCPT TO command is suppressed (treated as non-deliverable with a permanent or temporary error). This ruleset allows to require encryption or verification of the recipient's MTA even if the mail is somehow redirected to another host. For example, sending mail to *luke@endmail.org* may get redirected to a host named *death.star* and hence the tls_server ruleset won't apply. By introducing per recipient restrictions such attacks (e.g., via DNS spoofing) can be made impossible. See *cf/README* how this ruleset can be used.

#### 5.1.4.15. srv_features

The *srv_features* ruleset is called with the connecting client's host name when a client connects to sendmail. This ruleset should return **$#** followed by a list of options (single characters delimited by white space). If the return value starts with anything else it is silently ignored. Generally upper case characters turn off a feature while lower case characters turn it on. Option 'S' causes the server not to offer STARTTLS, which is useful to interact with MTAs/MUAs that have broken STARTTLS implementations by simply not offering it. ‘V’ turns off the request for a client certificate during the TLS handshake. Options ‘A’ and ‘P’ suppress SMTP AUTH and PIPELINING, respectively. ‘c’ is the equivalent to AuthOptions=p, i.e., it doesn’t permit mechanisms susceptible to simple passive attack (e.g., PLAIN, LOGIN), unless a security layer is active. Option ‘l’ requires SMTP AUTH for a connection. Options ’B’, ’D’, ’E’, and ’X’ suppress SMTP VERB, DSN, ETRN, and EXPN, respectively.

| | |
|---|---|
| A | Do not offer AUTH |
| a | Offer AUTH (default) |
| B | Do not offer VERB |
| b | Offer VERB (default) |
| C | Do not require security layer for plaintext AUTH (default) |
| c | Require security layer for plaintext AUTH |
| D | Do not offer DSN |
| d | Offer DSN (default) |
| E | Do not offer ETRN |
| e | Offer ETRN (default) |
| L | Do not require AUTH (default) |
| l | Require AUTH |
| P | Do not offer PIPELINING |
| p | Offer PIPELINING (default) |
| S | Do not offer STARTTLS |
| s | Offer STARTTLS (default) |
| V | Do not request a client certificate |
| v | Request a client certificate (default) |
| X | Do not offer EXPN |
| x | Offer EXPN (default) |

Note: the entries marked as ‘‘(default)’’ may require that some configuration has been made, e.g., SMTP AUTH is only available if properly configured. Moreover, many options can be changed on a global basis via other settings as explained in this document, e.g., via DaemonPortOptions.

The ruleset may return ‘$#temp’ to indicate that there is a temporary problem determining the correct features, e.g., if a map is unavailable. In that case, the SMTP server issues a temporary failure and does not accept email.

#### 5.1.4.16. try_tls

The *try_tls* ruleset is called when sendmail connects to another MTA. If the ruleset does resolve to the “error” mailer, sendmail does not try STARTTLS even if it is offered. This is useful to interact with MTAs that have broken STARTTLS implementations by simply not using it.

#### 5.1.4.17. authinfo

The *authinfo* ruleset is called when sendmail tries to authenticate to another MTA. It should return **$#** followed by a list of tokens that are used for SMTP AUTH. If the return value starts with anything else it is silently ignored. Each token is a tagged string of the form: "TDstring" (including the quotes), where

| | |
|---|---|
| T | Tag which describes the item |
| D | Delimiter: ':' simple text follows<br>'=' string is base64 encoded |
| string | Value of the item |

Valid values for the tag are:

| | |
|---|---|
| U | user (authorization) id |
| I | authentication id |
| P | password |
| R | realm |
| M | list of mechanisms delimited by spaces |

If this ruleset is defined, the option **DefaultAuthInfo** is ignored (even if the ruleset does not return a "useful" result).

#### 5.1.4.18. queuegroup

The *queuegroup* ruleset is used to map a recipient address to a queue group name. The input for the ruleset is a recipient address as specified by the SMTP RCPT command. The ruleset should return **$#** followed by the name of a queue group. If the return value starts with anything else it is silently ignored. See the section about "Queue Groups and Queue Directories" for further information.

#### 5.1.4.19. greet_pause

The *greet_pause* ruleset is used to specify the amount of time to pause before sending the initial SMTP 220 greeting. If any traffic is received during that pause, an SMTP 554 rejection response is given instead of the 220 greeting and all SMTP commands are rejected during that connection. This helps protect sites from open proxies and SMTP slammers. The ruleset should return **$#** followed by the number of milliseconds (thousandths of a second) to pause. If the return value starts with anything else or is not a number, it is silently ignored. Note: this ruleset is not invoked (and hence the feature is disabled) when the smtps (SMTP over SSL) is used, i.e., the *s* modifier is set for the daemon via **DaemonPortOptions**, because in this case the SSL handshake is performed before the greeting is sent.

### 5.1.5. 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 if not delivering via a named socket; this looks the name up in DNS to find alternate delivery sites.

The host name can also be provided as a dotted quad or an IPv6 address in square brackets; for example:

    [128.32.149.78]

or

    [IPv6:2002:c0a8:51d2::23f4]

This causes direct conversion of the numeric value to an 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. **N.B.:** *This is intended only for situations where you have a network firewall or other host that will do special processing for all your mail, 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.*

### 5.2. D — Define Macro

Macros are named with a single character or with a word in {braces}. The names ‘‘x’’ and ‘‘{x}’’ denote the same macro for every single character ‘‘x’’. Single character names may be selected from the entire ASCII set, but user-defined macros should be selected from the set of upper case letters only. Lower case letters and special symbols are used internally. Long names beginning with a lower case letter or a punctuation character are reserved for use by sendmail, so user-defined long macro names should begin with an upper case letter.

The syntax for macro definitions is:

**D***x val*

where *x* is the name of the macro (which may be a single character or a word in braces) and *val* is the value it should have. There should be no spaces given that do not actually belong in the macro value.

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 text1 $| text2 $.

This interpolates *text1* if the macro **$x** is set and non-null, and *text2* otherwise. The “else” (**$|**) clause may be omitted.

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[16], 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. These macros are:

$a The origination date in RFC 822 format. This is extracted from the Date: line.

$b The current date in RFC 822 format.

$c The hop count. This is a count of the number of Received: lines plus the value of the **–h** command line flag.

$d The current date in UNIX (ctime) format.

$e† (Obsolete; use SmtpGreetingMessage option instead.) The SMTP entry message. This is printed out when SMTP starts up. The first word must be the **$j** macro as specified by RFC 821. Defaults to “$j Sendmail $v ready at $b”. Commonly redefined to include the configuration version number, e.g., “$j Sendmail $v/$Z ready at $b”

$f The envelope sender (from) address.

$g The sender address relative to the recipient. For example, if **$f** is “foo”, **$g** will be “host!foo”, “foo@host.domain”, or whatever is appropriate for the receiving mailer.

[16]As of version 8.6, all of these macros have reasonable defaults. Previous versions required that they be defined.

$h The recipient host. This is set in ruleset 0 from the $@ field of a parsed address.

$i The queue id, e.g., "f344MXxp018717".

$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 information can find it automatically.

$k The UUCP node name (from the uname system call).

$l† (Obsolete; use UnixFromLine option instead.) The format of the UNIX from line. Unless you have changed the UNIX 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".

$o† (Obsolete: use OperatorChars option instead.) The set of "operators" in addresses. A list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if "@" were in the **$o** macro, then the input "a@b" would be scanned as three tokens: "a," "@," and "b." Defaults to ".:@[]", 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)$.", corresponding 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. Set from the **−p** command line flag or by the SMTP server code.

$s Sender's host name. Set from the **−p** command line flag or by the SMTP server code (in which case it is set to the EHLO/HELO parameter).

$t A numeric representation of the current time in the format YYYYMMDDHHmm (4 digit year 1900-9999, 2 digit month 01-12, 2 digit day 01-31, 2 digit hours 00-23, 2 digit minutes 00-59).

$u The recipient user.

$v The version number of the *sendmail* binary.

$w‡ The hostname of this site. This is 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.

$_ The validated sender address. See also **${client_resolve}**.

${addr_type}
The type of the address which is currently being rewritten. This macro contains up to three characters, the first is either 'e' or 'h' for envelope/header address, the second is a space, and the third is either 's' or 'r' for sender/recipient address.

${alg_bits}
The maximum keylength (in bits) of the symmetric encryption algorithm used for a TLS connection. This may be less than the effective keylength, which is stored in **${cipher_bits}**, for

''export controlled'' algorithms.

${auth_authen}
: The client's authentication credentials as determined by authentication (only set if successful). The format depends on the mechanism used, it might be just 'user', or 'user@realm', or something similar (SMTP AUTH only).

${auth_author}
: The authorization identity, i.e. the AUTH= parameter of the SMTP MAIL command if supplied.

${auth_type}
: The mechanism used for SMTP authentication (only set if successful).

${auth_ssf}
: The keylength (in bits) of the symmetric encryption algorithm used for the security layer of a SASL mechanism.

${bodytype}
: The message body type (7BIT or 8BITMIME), as determined from the envelope.

${cert_issuer}
: The DN (distinguished name) of the CA (certificate authority) that signed the presented certificate (the cert issuer) (STARTTLS only).

${cert_md5}
: The MD5 hash of the presented certificate (STARTTLS only).

${cert_subject}
: The DN of the presented certificate (called the cert subject) (STARTTLS only).

${cipher}
: The cipher suite used for the connection, e.g., EDH-DSS-DES-CBC3-SHA, EDH-RSA-DES-CBC-SHA, DES-CBC-MD5, DES-CBC3-SHA (STARTTLS only).

${cipher_bits}
: The effective keylength (in bits) of the symmetric encryption algorithm used for a TLS connection.

${client_addr}
: The IP address of the SMTP client. IPv6 addresses are tagged with "IPv6:" before the address. Defined in the SMTP server only.

${client_connections}
: The number of open connections in the SMTP server for the client IP address.

${client_flags}
: The flags specified by the Modifier= part of **ClientPortOptions** where flags are separated from each other by spaces and upper case flags are doubled. That is, Modifier=hA will be represented as "h AA" in **${client_flags}**, which is required for testing the flags in rulesets.

${client_name}
: The host name of the SMTP client. This may be the client's bracketed IP address in the form [ nnn.nnn.nnn.nnn ] for IPv4 and [ IPv6:nnnn:...:nnnn ] for IPv6 if the client's IP address is not resolvable, or if it is resolvable but the IP address of the resolved hostname doesn't match the original IP address. Defined in the SMTP server only. See also **${client_resolve}**.

${client_port}
: The port number of the SMTP client. Defined in the SMTP server only.

${client_ptr}
: The result of the PTR lookup for the client IP address. Note: this is the same as **${client_name}** if and only if **${client_resolve}** is OK. Defined in the SMTP server only.

${client_rate}
The number of incoming connections for the client IP address over the time interval specified by ConnectionRateWindowSize.

${client_resolve}
Holds the result of the resolve call for **${client_name}**. Possible values are:

| | |
|---|---|
| OK | resolved successfully |
| FAIL | permanent lookup failure |
| FORGED | forward lookup doesn't match reverse lookup |
| TEMP | temporary lookup failure |

Defined in the SMTP server only. *sendmail* performs a hostname lookup on the IP address of the connecting client. Next the IP addresses of that hostname are looked up. If the client IP address does not appear in that list, then the hostname is maybe forged. This is reflected as the value FORGED for **${client_resolve}** and it also shows up in **$_** as "(may be forged)".

${cn_issuer}
The CN (common name) of the CA that signed the presented certificate (STARTTLS only). Note: if the CN cannot be extracted properly it will be replaced by one of these strings based on the encountered error:

| | |
|---|---|
| BadCertificateContainsNUL | CN contains a NUL character |
| BadCertificateTooLong | CN is too long |
| BadCertificateUnknown | CN could not be extracted |

In the last case, some other (unspecific) error occurred.

${cn_subject}
The CN (common name) of the presented certificate (STARTTLS only). See **${cn_issuer}** for possible replacements.

${currHeader}
Header value as quoted string (possibly truncated to **MAXNAME**). This macro is only available in header check rulesets.

${daemon_addr}
The IP address the daemon is listening on for connections.

${daemon_family}
The network family if the daemon is accepting network connections. Possible values include "inet", "inet6", "iso", "ns", "x.25"

${daemon_flags}
The flags for the daemon as specified by the Modifier= part of **DaemonPortOptions** whereby the flags are separated from each other by spaces, and upper case flags are doubled. That is, Modifier=Ea will be represented as "EE a" in **${daemon_flags}**, which is required for testing the flags in rulesets.

${daemon_info}
Some information about a daemon as a text string. For example, "SMTP+queueing@00:30:00".

${daemon_name}
The name of the daemon from **DaemonPortOptions** Name= suboption. If this suboption is not set, "Daemon#", where # is the daemon number, is used.

${daemon_port}
The port the daemon is accepting connection on. Unless **DaemonPortOptions** is set, this will most likely be "25".

${deliveryMode}
The current delivery mode sendmail is using. It is initially set to the value of the

**DeliveryMode** option.

${envid}
The envelope id parameter (ENVID=) passed to sendmail as part of the envelope.

${hdrlen}
The length of the header value which is stored in ${currHeader} (before possible truncation). If this value is greater than or equal to **MAXNAME** the header has been truncated.

${hdr_name}
The name of the header field for which the current header check ruleset has been called. This is useful for a default header check ruleset to get the name of the header; the macro is only available in header check rulesets.

${if_addr}
The IP address of the interface of an incoming connection unless it is in the loopback net. IPv6 addresses are tagged with "IPv6:" before the address.

${if_addr_out}
The IP address of the interface of an outgoing connection unless it is in the loopback net. IPv6 addresses are tagged with "IPv6:" before the address.

${if_family}
The IP family of the interface of an incoming connection unless it is in the loopback net.

${if_family_out}
The IP family of the interface of an outgoing connection unless it is in the loopback net.

${if_name}
The hostname associated with the interface of an incoming connection. This macro can be used for SmtpGreetingMessage and HReceived for virtual hosting. For example:

```
O SmtpGreetingMessage=$?{if_name}${if_name}$|$j$. MTA
```

${if_name_out}
The name of the interface of an outgoing connection.

${load_avg}
The current load average.

${mail_addr}
The address part of the resolved triple of the address given for the SMTP MAIL command. Defined in the SMTP server only.

${mail_host}
The host from the resolved triple of the address given for the SMTP MAIL command. Defined in the SMTP server only.

${mail_mailer}
The mailer from the resolved triple of the address given for the SMTP MAIL command. Defined in the SMTP server only.

${msg_id}
The value of the Message-Id: header.

${msg_size}
The value of the SIZE= parameter, i.e., usually the size of the message (in an ESMTP dialogue), before the message has been collected, thereafter the message size as computed by *sendmail* (and can be used in check_compat).

${nbadrcpts}
The number of bad recipients for a single message.

${nrcpts}
: The number of validated recipients for a single message. Note: since recipient validation happens after *check_rcpt* has been called, the value in this ruleset is one less than what might be expected.

${ntries}
: The number of delivery attempts.

${opMode}
: The current operation mode (from the **−b** flag).

${quarantine}
: The quarantine reason for the envelope, if it is quarantined.

${queue_interval}
: The queue run interval given by the **−q** flag. For example, **−q30m** would set **${queue_interval}** to “00:30:00”.

${rcpt_addr}
: The address part of the resolved triple of the address given for the SMTP RCPT command. Defined in the SMTP server only after a RCPT command.

${rcpt_host}
: The host from the resolved triple of the address given for the SMTP RCPT command. Defined in the SMTP server only after a RCPT command.

${rcpt_mailer}
: The mailer from the resolved triple of the address given for the SMTP RCPT command. Defined in the SMTP server only after a RCPT command.

${server_addr}
: The address of the server of the current outgoing SMTP connection. For LMTP delivery the macro is set to the name of the mailer.

${server_name}
: The name of the server of the current outgoing SMTP or LMTP connection.

${time}
: The output of the *time*(3) function, i.e., the number of seconds since 0 hours, 0 minutes, 0 seconds, January 1, 1970, Coordinated Universal Time (UTC).

${tls_version}
: The TLS/SSL version used for the connection, e.g., TLSv1, SSLv3, SSLv2; defined after STARTTLS has been used.

${total_rate}
: The total number of incoming connections over the time interval specified by ConnectionRateWindowSize.

${verify}
: The result of the verification of the presented cert; only defined after STARTTLS has been used (or attempted). Possible values are:

| | |
|---|---|
| OK | verification succeeded. |
| NO | no cert presented. |
| NOT | no cert requested. |
| FAIL | cert presented but could not be verified, e.g., the signing CA is missing. |
| NONE | STARTTLS has not been performed. |
| TEMP | temporary error occurred. |
| PROTOCOL | some protocol error occurred at the ESMTP level (not TLS). |
| SOFTWARE | STARTTLS handshake failed, which is a fatal error for this session, the e-mail will be queued. |

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 (ctime) format.

The macros **$w**, **$j**, and **$m** are set to the identity of this host. *Sendmail* tries to find the fully qualified name of the host if at all possible; it does this by calling *gethostname*(2) to get the current hostname and then passing that to *gethostbyname*(3) which is supposed to return the canonical version of that host name.[17] Assuming this is successful, **$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 **$j**. If the canonification is not successful, it is imperative that the config file set **$j** to the fully qualified domain name[18].

The **$f** macro is the id of the sender as originally determined; when mailing to a specific host the **$g** macro is set to the address of the sender *relative to the recipient.* For example, if I send to "bollard@matisse.CS.Berkeley.EDU" from the machine "vangogh.CS.Berkeley.EDU" the **$f** macro will be "eric" and the **$g** macro will be "eric@vangogh.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 flag to *sendmail*. It can be defined in the NAME environment variable. The third choice is the value of the "Full-Name:" line in the header if it exists, and the fourth 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 **$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the **$@** and **$:** part of the rewriting rules, respectively.

The **$p** and **$t** macros are used to create unique strings (e.g., 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 been proven extremely useful for debugging.

The **$c** field is set to the "hop count," i.e., 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.

[17]For example, on some systems *gethostname* might return "foo" which would be mapped to "foo.bar.com" by *gethostbyname*.

[18]Older versions of sendmail didn't pre-define **$j** at all, so up until 8.6, config files *always* had to define **$j**.

The **$r** and **$s** fields are set to the protocol used to communicate with *sendmail* and the sending hostname. They can be set together using the **−p** command line flag or separately using the **−M** or **−oM** flags.

The **$_** is set to a validated sender host name. If the sender is running an RFC 1413 compliant IDENT server and the receiver has the IDENT protocol turned on, it will include the user name on that host.

The **${client_name}**, **${client_addr}**, and **${client_port}** macros are set to the name, address, and port number of the SMTP client who is invoking *sendmail* as a server. These can be used in the *check_** rulesets (using the **$&** deferred evaluation form, of course!).

### 5.3. 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 does 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 are named as a single letter or a word in {braces}. Class names beginning with lower case letters and special characters are reserved for system use. Classes defined in config files may be given names from the set of upper case letters for short names or beginning with an upper case letter for long names.

The syntax is:

**C***c phrase1 phrase2...*
**F***c file*
**F***c |program*
**F***c [mapkey]@mapclass:mapspec*

The first form defines the class *c* to match any of the named words. If *phrase1* or *phrase2* is another class, e.g., *$=S*, the contents of class *S* are added to class *c*. It is permissible to split them among multiple lines; for example, the two forms:

CHmonet ucbmonet

and

CHmonet
CHucbmonet

are equivalent. The ''F'' forms read the elements of the class *c* from the named *file*, *program*, or *map specification*. Each element should be listed on a separate line. To specify an optional file, use ''−o'' between the class name and the file name, e.g.,

Fc −o /path/to/file

If the file can't be used, *sendmail* will not complain but silently ignore it. The map form should be an optional map key, an at sign, and a map class followed by the specification for that map. Examples include:

F{VirtHosts}@ldap:−k (&(objectClass=virtHosts)(host=*)) −v host
F{MyClass}foo@hash:/etc/mail/classes

will fill the class **$={VirtHosts}** from an LDAP map lookup and **$={MyClass}** from a hash database map lookup of the **foo**. There is also a built-in schema that can be accessed by only specifying:

F{*ClassName*}@LDAP

This will tell sendmail to use the default schema:

```
−k (&(objectClass=sendmailMTAClass)
   (sendmailMTAClassName=ClassName)
   (|(sendmailMTACluster=${sendmailMTACluster})
    (sendmailMTAHost=$j)))
−v sendmailMTAClassValue
```

Note that the lookup is only done when sendmail is initially started.

Elements of classes can be accessed in rules using **$=** or **$~**. The **$~** (match entries not in class) only matches a single word; multi-word entries in the class are ignored in this context.

Some classes have internal meaning to *sendmail*:

$=e contains the Content-Transfer-Encodings that can be 8→7 bit encoded. It is predefined to contain “7bit”, “8bit”, and “binary”.

$=k set to be the same as **$k**, that is, the UUCP node name.

$=m set to the set of domains by which this host is known, initially just **$m**.

$=n can be set to the set of MIME body types that can never be eight to seven bit encoded. It defaults to “multipart/signed”. Message types “message/*” and “multipart/*” are never encoded directly. Multipart messages are always handled recursively. The handling of message/* messages are controlled by class **$=s**.

$=q A set of Content-Types that will never be encoded as base64 (if they have to be encoded, they will be encoded as quoted-printable). It can have primary types (e.g., “text”) or full types (such as “text/plain”).

$=s contains the set of subtypes of message that can be treated recursively. By default it contains only “rfc822”. Other “message/*” types cannot be 8→7 bit encoded. If a message containing eight bit data is sent to a seven bit host, and that message cannot be encoded into seven bits, it will be stripped to 7 bits.

$=t set to the set of trusted users by the **T** configuration line. If you want to read trusted users from a file, use **Ft***/file/name*.

$=w set to be the set of all names this host is known by. This can be used to match local hostnames.

${persistentMacros}
set to the macros that should be saved across queue runs. Care should be taken when adding macro names to this class.

*Sendmail* can be compiled to allow a *scanf*(3) string on the **F** line. This lets you do simplistic parsing of text files. For example, to read all the user names in your system */etc/passwd* file into a class, use

```
FL/etc/passwd %[^:]
```

which reads every line up to the first colon.

### 5.4. M — Define Mailer

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

**M***name*, {*field*=*value* }*

where *name* is the name of the mailer (used internally only) and the “field=name” pairs define attributes of the mailer. Fields are:

| | |
|---|---|
| Path | The pathname of the mailer |
| Flags | Special flags for this mailer |
| Sender | Rewriting set(s) for sender addresses |
| Recipient | Rewriting set(s) for recipient addresses |
| recipients | Maximum number of recipients per connection |
| Argv | An argument vector to pass to this mailer |
| Eol | The end-of-line string for this mailer |
| Maxsize | The maximum message length to this mailer |
| maxmessages | The maximum message deliveries per connection |
| Linelimit | The maximum line length in the message body |
| Directory | The working directory for the mailer |
| Userid | The default user and group id to run as |
| Nice | The nice(2) increment for the mailer |
| Charset | The default character set for 8-bit characters |
| Type | Type information for DSN diagnostics |
| Wait | The maximum time to wait for the mailer |
| Queuegroup | The default queue group for the mailer |
| / | The root directory for the mailer |

Only the first character of the field name is checked (it's case-sensitive).

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. Flags marked with † are not interpreted by the *sendmail* binary; these are the conventionally used to correlate to the flags portion of the **H** line. Flags marked with ‡ apply to the mailers for the sender address rather than the usual recipient mailers.

a Run Extended SMTP (ESMTP) protocol (defined in RFCs 1869, 1652, and 1870). This flag defaults on if the SMTP greeting message includes the word "ESMTP".

A Look up the user (address) part of the resolved mailer triple, in the alias database. Normally this is only set for local mailers.

b Force a blank line on the end of a message. This is intended to work around some stupid versions of /bin/mail that require a blank line, but do not provide it themselves. It would not normally be used on network mail.

B Strip leading backslashes (\) off of the address; this is a subset of the functionality of the **s** flag.

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. This strips addresses of the form "Phrase <address>" or "address (Comment)" down to just "address".

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 will have the "@domain" clause from the sender envelope address tacked on. This allows mail with headers of the form:

    From: usera@hosta
    To: userb@hostb, userc

to be rewritten as:

    From: usera@hosta
    To: userb@hostb, userc@hosta

automatically. However, it doesn't really work reliably.

d Do not include angle brackets around route-address syntax addresses. This is useful on mailers that are going to pass addresses to a shell that might interpret angle brackets as I/O redirection. However, it does not protect against other shell metacharacters. Therefore, passing addresses to a shell should not be considered secure.

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 will occur during a queue run. See also option **HoldExpensive**.

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 (i.e., the mailer will give 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 (e.g., error messages) using the null return address as required by RFC 1123. However, some mailers don’t 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 Upper case should be preserved in host names (the $@ portion of the mailer triplet resolved from ruleset 0) for this mailer.

i Do User Database rewriting on envelope sender address.

I This mailer will be speaking SMTP to another *sendmail* — as such it can use special protocol features. This flag should not be used except for debugging purposes because it uses **VERB** as SMTP command.

j Do User Database rewriting on recipients as well as senders.

k Normally when *sendmail* connects to a host via SMTP, it checks to make sure that this isn’t accidently the same host name as might happen if *sendmail* is misconfigured or if a long-haul network interface is set in loopback mode. This flag disables the loopback check. It should only be used under very unusual circumstances.

K Currently unimplemented. Reserved for chunking.

l This mailer is local (i.e., final delivery will be 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 will be repeated as necessary for all qualifying users. Removing this flag can defeat duplicate supression on a remote site as each recipient is sent in a separate transaction.

M† This mailer wants a “Message-Id:” header line.

n Do not insert a UNIX-style “From” line on the front of the message.

o Always run as the owner of the recipient mailbox. Normally *sendmail* runs as the sender for locally generated mail or as “daemon” (actually, the user specified in the **u** option) when delivering network mail. The normal behavior is required by most local mailers, which will not allow the envelope sender address to be set unless the mailer is running as daemon. This flag is ignored if the **S** flag is set.

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 wants a “Return-Path:” line.

q When an address that resolves to this mailer is verified (SMTP VRFY command), generate 250 responses instead of 252 responses. This will imply that the address is local.

r Same as **f**, but sends a **−r** flag.

R Open SMTP connections from a “secure” port. Secure ports aren’t (secure, that is) except on UNIX machines, so it is unclear that this adds anything. *sendmail* must be running as root to be able to use this flag.

s Strip quote characters (" and \) off of the address before calling the mailer.

S Don’t reset the userid 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. If the **U=** field is also specified, this flag causes the effective user id to be set to that user.

u Upper case should be preserved in user names for this mailer. Standards require preservation of case in the local part of addresses, except for those address for which your system accepts responsibility. RFC 2142 provides a long list of addresses which should be case insensitive. If you use this flag, you may be violating RFC 2142. Note that postmaster is always treated as a case insensitive address regardless of this flag.

U This mailer wants UUCP-style “From” lines with the ugly “remote from <host>” on the end.

w The user must have a valid account on this machine, i.e., *getpwnam* must succeed. If not, the mail is bounced. See also the **MailBoxDatabase** option. This is required to get “.forward” capability.

W Ignore long term host status information (see Section "Persistent Host Status Information").

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 will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.

z Run Local Mail Transfer Protocol (LMTP) between *sendmail* and the local mailer. This is a variant on SMTP defined in RFC 2033 that is specifically designed for delivery to a local mailbox.

Z Apply DialDelay (if set) to this mailer.

0 Don’t look up MX records for hosts sent via SMTP/LMTP. Do not apply **FallbackMXhost** either.

1 Don’t send null characters (’\0’) to this mailer.

2 Don’t use ESMTP even if offered; this is useful for broken systems that offer ESMTP but fail on EHLO (without recovering when HELO is tried next).

3 Extend the list of characters converted to =XX notation when converting to Quoted-Printable to include those that don’t map cleanly between ASCII and EBCDIC. Useful if you have IBM mainframes on site.

5 If no aliases are found for this address, pass the address through ruleset 5 for possible alternate resolution. This is intended to forward the mail to an alternate delivery spot.

6 Strip headers to seven bits.

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, since the eighth bit was stripped on input. Note that this option will only impact messages that didn’t have 8→7 bit MIME conversions performed.

8 If set, it is acceptable to send eight bit data to this mailer; the usual attempt to do 8→7 bit MIME conversions will be bypassed.

9 If set, do *limited* 7→8 bit MIME conversions. These conversions are limited to text/plain data.

: Check addresses to see if they begin “:include:”; if they do, convert them to the “*include*” mailer.

| Check addresses to see if they begin with a '|'; if they do, convert them to the "prog" mailer.

/ Check addresses to see if they begin with a '/'; if they do, convert them to the "*file*" mailer.

@ Look up addresses in the user database.

% Do not attempt delivery on initial receipt of a message or on queue runs unless the queued message is selected using one of the -qI/-qR/-qS queue run modifiers or an ETRN request.

Configuration files prior to level 6 assume the 'A', 'w', '5', ':', '|', '/', and '@' options on the mailer named "local".

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, or an enhanced error code as described in RFC 1893, *Enhanced Mail System Status Codes.* 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 rulesets 0, 5, or one of the check_* rulesets. The host field can also contain the special token **quarantine** which instructs sendmail to quarantine the current message.

The mailer with the special name "discard" causes any mail sent to it to be discarded but otherwise treated as though it were successfully delivered. This mailer cannot be used in ruleset 0, only in the various address checking rulesets.

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=lsoDq9, T=DNS/RFC822/X-Unix, A=sh −c $u
M*file*, P=[FILE], F=lsDFMPEouq9, T=DNS/RFC822/X-Unix, A=FILE $u
M*include*, P=/dev/null, F=su, A=INCLUDE $u
```

Builtin pathnames are [FILE] and [IPC], the former is used for delivery to files, the latter for delivery via interprocess communication. For mailers that use [IPC] as pathname the argument vector (A=) must start with TCP or FILE for delivery via a TCP or a Unix domain socket. If TCP is used, the second argument must be the name of the host to contact. Optionally a third argument can be used to specify a port, the default is smtp (port 25). If FILE is used, the second argument must be the name of the Unix domain socket.

If the argument vector does not contain $u then *sendmail* will speak SMTP (or LMTP if the mailer flag z is specified) to the mailer.

If no Eol field is defined, then the default is "\r\n" for SMTP mailers and "\n" of others.

The Sender and Recipient rewriting sets may either be a simple ruleset id or may be two ids separated by a slash; if so, the first rewriting set is applied to envelope addresses and the second is applied to headers. Setting any value to zero disables corresponding mailer-specific rewriting.

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, since some shells (such as *csh*) refuse to execute if they cannot read the current directory. Since the queue directory is not normally readable by unprivileged users *csh* scripts as recipients can fail.

The Userid specifies the default user and group id to run as, overriding the **DefaultUser** option (q.v.). If the **S** mailer flag is also specified, this user and group will be set as the effective uid and gid for the process. This may be given as *user:group* to set both the user and group id; either may be an integer or a symbolic name to be looked up in the *passwd* and *group* files respectively. If only a symbolic user name is specified, the group id in the *passwd* file for that user is used as the group id.

The Charset field is used when converting a message to MIME; this is the character set used in the Content-Type: header. If this is not set, the **DefaultCharset** option is used, and if that is not set, the value "unknown-8bit" is used. **WARNING:** this field applies to the sender's mailer, not the recipient's mailer. For example, if the envelope sender address lists an address on the local network and the recipient is on an external network, the character set will be set from the Charset= field for the local network mailer, not that of the external network mailer.

The Type= field sets the type information used in MIME error messages as defined by RFC 1894. It is actually three values separated by slashes: the MTA-type (that is, the description of how hosts are named), the address type (the description of e-mail addresses), and the diagnostic type (the description of error diagnostic codes). Each of these must be a registered value or begin with "X−". The default is "dns/rfc822/smtp".

The m= field specifies the maximum number of messages to attempt to deliver on a single SMTP or LMTP connection. The default is infinite.

The r= field specifies the maximum number of recipients to attempt to deliver in a single envelope. It defaults to 100.

The /= field specifies a new root directory for the mailer. The path is macro expanded and then passed to the "chroot" system call. The root directory is changed before the Directory field is consulted or the uid is changed.

The Wait= field specifies the maximum time to wait for the mailer to return after sending all data to it. This applies to mailers that have been forked by *sendmail*.

The Queuegroup= field specifies the default queue group in which received mail should be queued. This can be overridden by other means as explained in section ''Queue Groups and Queue Directories''.

## 5.5. H — Define Header

The format of the header lines that *sendmail* inserts into the message are defined by the **H** line. The syntax of this line is one of the following:

> **H***hname***:** *htemplate*
>
> **H**[**?***mflags***?**]*hname***:** *htemplate*
>
> **H**[**?$***{macro}***?**]*hname***:** *htemplate*

Continuation lines in this spec 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 a *${macro}* (surrounded by question marks) is specified, the header will be automatically output if the macro is set. The macro may be set using any of the normal methods, including using the **macro** storage map in a ruleset. If one of these headers is in the input it is reflected to the output regardless of these flags or macros. Notice: If a *${macro}* is used to set a header, then it is useful to add that macro to class *$={persistentMacros}* which consists of the macros that should be saved across queue runs.

Some headers have special semantics that will be described later.

A secondary syntax allows validation of headers as they are being read. To enable validation, use:

**H***Header***: $>***Ruleset*
**H***Header***: $>+***Ruleset*

The indicated *Ruleset* is called for the specified *Header*, and can return **$#error** to reject or quarantine the message or **$#discard** to discard the message (as with the other **check_*** rulesets). The ruleset receives the header field-body as argument, i.e., not the header field-name; see also ${hdr_name} and ${currHeader}. The header is treated as a structured field, that is, text in parentheses is deleted before processing, unless the second form **$>+** is used. Note: only one ruleset can be associated with a header; *sendmail* will silently ignore multiple entries.

For example, the configuration lines:

```
HMessage-Id: $>CheckMessageId

SCheckMessageId
R< $+ @ $+ >    $@ OK
R$*             $#error $: Illegal Message-Id header
```

would refuse any message that had a Message-Id: header of any of the following forms:

```
Message-Id: <>
Message-Id: some text
Message-Id: <legal text@domain> extra crud
```

A default ruleset that is called for headers which don't have a specific ruleset defined for them can be specified by:

**H*: $>***Ruleset*

or

**H*: $>+***Ruleset*

## 5.6. O — Set Option

There are a number of global options that can be set from a configuration file. Options are represented by full words; some are also representable as single characters for back compatibility. The syntax of this line is:

**O** *option***=***value*

This sets option *option* to be *value*. Note that there *must* be a space between the letter 'O' and the name of the option. An older version is:

**O***o value*

where the option *o* is a single character. Depending on the option, *value* may be a string, an integer, a boolean (with legal values "t", "T", "f", or "F"; the default is TRUE), or a time interval.

All filenames used in options should be absolute paths, i.e., starting with '/'. Relative filenames most likely cause surprises during operation (unless otherwise noted).

The options supported (with the old, one character names in brackets) are:

AliasFile=*spec, spec, ...*

[A] Specify possible alias file(s). Each *spec* should be in the format ''*class***:** *info*'' where *class***:** is optional and defaults to ''implicit''. Note that *info* is required for all *class*es except "ldap". For the "ldap" class, if *info* is not specified, a default *info* value is used as follows:

−k (&(objectClass=sendmailMTAAliasObject)
(sendmailMTAAliasName=aliases)
(|(sendmailMTACluster=${sendmailMTACluster})
(sendmailMTAHost=$j))
(sendmailMTAKey=%0))
−v sendmailMTAAliasValue

Depending on how *sendmail* is compiled, valid classes are "implicit" (search through a compiled-in list of alias file types, for back compatibility), "hash" (if NEWDB is specified), "btree" (if NEWDB is specified), "dbm" (if NDBM is specified), "stab" (internal symbol table — not normally used unless you have no other database lookup), "sequence" (use a sequence of maps previously declared), "ldap" (if LDAPMAP is specified), or "nis" (if NIS is specified). If a list of *spec*s are provided, *sendmail* searches them in order.

AliasWait=*timeout*
: [a] If set, wait up to *timeout* (units default to minutes) for an "@:@" entry to exist in the alias database before starting up. If it does not appear in the *timeout* interval issue a warning.

AllowBogusHELO
: [no short name] If set, allow HELO SMTP commands that don't include a host name. Setting this violates RFC 1123 section 5.2.5, but is necessary to interoperate with several SMTP clients. If there is a value, it is still checked for legitimacy.

AuthMaxBits=*N*
: [no short name] Limit the maximum encryption strength for the security layer in SMTP AUTH (SASL). Default is essentially unlimited. This allows to turn off additional encryption in SASL if STARTTLS is already encrypting the communication, because the existing encryption strength is taken into account when choosing an algorithm for the security layer. For example, if STARTTLS is used and the symmetric cipher is 3DES, then the the keylength (in bits) is 168. Hence setting **AuthMaxBits** to 168 will disable any encryption in SASL.

AuthMechanisms
: [no short name] List of authentication mechanisms for AUTH (separated by spaces). The advertised list of authentication mechanisms will be the intersection of this list and the list of available mechanisms as determined by the Cyrus SASL library. If STARTTLS is active, EXTERNAL will be added to this list. In that case, the value of {cert_subject} is used as authentication id.

AuthOptions
: [no short name] List of options for SMTP AUTH consisting of single characters with intervening white space or commas.

| | |
|---|---|
| A | Use the AUTH= parameter for the MAIL FROM command only when authentication succeeded. This can be used as a workaround for broken MTAs that do not implement RFC 2554 correctly. |
| a | protection from active (non-dictionary) attacks during authentication exchange. |
| c | require mechanisms which pass client credentials, and allow mechanisms which can pass credentials to do so. |
| d | don't permit mechanisms susceptible to passive dictionary attack. |
| f | require forward secrecy between sessions (breaking one won't help break next). |
| m | require mechanisms which provide mutual authentication (only available if using Cyrus SASL v2 or later). |
| p | don't permit mechanisms susceptible to simple passive attack (e.g., PLAIN, LOGIN), unless a security layer is active. |
| y | don't permit mechanisms that allow anonymous login. |

The first option applies to sendmail as a client, the others to a server. Example:

```
O AuthOptions=p,y
```

would disallow ANONYMOUS as AUTH mechanism and would allow PLAIN and LOGIN only if a security layer (e.g., provided by STARTTLS) is already active. The options 'a', 'c', 'd', 'f', 'p', and 'y' refer to properties of the selected SASL mechanisms. Explanations of these properties can be found in the Cyrus SASL documentation.

AuthRealm
: [no short name] The authentication realm that is passed to the Cyrus SASL library. If no realm is specified, **$j** is used.

BadRcptThrottle=*N*
: [no short name] If set and the specified number of recipients in a single SMTP transaction have been rejected, sleep for one second after each subsequent RCPT command in that transaction.

BlankSub=*c*
: [B] Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to space (i.e., no change is made).

CACertPath
: [no short name] Path to directory with certificates of CAs. This directory directory must contain the hashes of each CA certificate as filenames (or as links to them).

CACertFile
: [no short name] File containing one or more CA certificates; see section about STARTTLS for more information.

CheckAliases
: [n] Validate the RHS of aliases when rebuilding the alias database.

CheckpointInterval=*N*
: [C] 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.

ClassFactor=*fact*
: [z] 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: will be favored. Defaults to 1800.

ClientCertFile [no short name] File containing the certificate of the client, i.e., this certificate is used when *sendmail* acts as client (for STARTTLS).

ClientKeyFile [no short name] File containing the private key belonging to the client certificate (for STARTTLS if *sendmail* runs as client).

ClientPortOptions=*options*
[O] Set client SMTP options. The options are *key=value* pairs separated by commas. Known keys are:

| | |
|---|---|
| Port | Name/number of source port for connection (defaults to any free port) |
| Addr | Address mask (defaults INADDR_ANY) |
| Family | Address family (defaults to INET) |
| SndBufSize | Size of TCP send buffer |
| RcvBufSize | Size of TCP receive buffer |
| Modifier | Options (flags) for the client |

The *Addr*ess mask may be a numeric address in IPv4 dot notation or IPv6 colon notation or a network name. Note that if a network name is specified, only the first IP address returned for it will be used. This may cause indeterminate behavior for network names that resolve to multiple addresses. Therefore, use of an address is recommended. *Modifier* can be the following character:

| | |
|---|---|
| h | use name of interface for HELO command |
| A | don't use AUTH when sending e-mail |
| S | don't use STARTTLS when sending e-mail |

If "h" is set, the name corresponding to the outgoing interface address (whether chosen via the Connection parameter or the default) is used for the HELO/EHLO command. However, the name must not start with a square bracket and it must contain at least one dot. This is a simple test whether the name is not an IP address (in square brackets) but a qualified hostname. Note that multiple ClientPortOptions settings are allowed in order to give settings for each protocol family (e.g., one for Family=inet and one for Family=inet6). A restriction placed on one family only affects outgoing connections on that particular family.

ColonOkInAddr [no short name] If set, colons are acceptable in e-mail addresses (e.g., "host:user"). If not set, colons indicate the beginning of a RFC 822 group construct ("groupname: member1, member2, ... memberN;"). Doubled colons are always acceptable ("nodename::user") and proper route-addr nesting is understood ("<@relay:user@host>"). Furthermore, this option defaults on if the configuration version level is less than 6 (for back compatibility). However, it must be off for full compatibility with RFC 822.

ConnectionCacheSize=*N*
[k] 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. Since this consumes file descriptors, the connection cache should be kept small: 4 is probably a practical maximum.

ConnectionCacheTimeout=*timeout*
[K] 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 RSET 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.

ConnectOnlyTo=*address*
: [no short name] This can be used to override the connection address (for testing purposes).

ConnectionRateThrottle=*N*
: [no short name] If set to a positive value, allow no more than *N* incoming connections in a one second period per daemon. This is intended to flatten out peaks and allow the load average checking to cut in. Defaults to zero (no limits).

ConnectionRateWindowSize=*N*
: [no short name] Define the length of the interval for which the number of incoming connections is maintained. The default is 60 seconds.

ControlSocketName=*name*
: [no short name] Name of the control socket for daemon management. A running *sendmail* daemon can be controlled through this named socket. Available commands are: *help, mstat, restart, shutdown,* and *status.* The *status* command returns the current number of daemon children, the maximum number of daemon children, the free disk space (in blocks) of the queue directory, and the load average of the machine expressed as an integer. If not set, no control socket will be available. Solaris and pre-4.4BSD kernel users should see the note in sendmail/README .

CRLFile=*name*
: [no short name] Name of file that contains certificate revocation status, useful for X.509v3 authentication. CRL checking requires at least OpenSSL version 0.9.7. Note: if a CRLFile is specified but the file is unusable, STARTTLS is disabled.

DHParameters
: Possible values are:

| | |
|---|---|
| 5 | use 512 bit prime |
| 1 | use 1024 bit prime |
| none | do not use Diffie-Hellman |
| NAME | load prime from file |

This is only required if a ciphersuite containing DSA/DH is used. If “5” is selected, then precomputed, fixed primes are used. This is the default for the client side. If “1” is selected, then prime values are computed during startup. This is the default for the server side. Note: this operation can take a significant amount of time on a slow machine (several seconds), but it is only done once at startup. If “none” is selected, then TLS ciphersuites containing DSA/DH cannot be used. If a file name is specified (which must be an absolute path), then the primes are read from it.

DaemonPortOptions=*options*
: [O] Set server SMTP options. Each instance of **DaemonPortOptions** leads to an additional incoming socket. The options are *key=value* pairs. Known keys are:

| | |
|---|---|
| Name | User-definable name for the daemon (defaults to "Daemon#") |
| Port | Name/number of listening port (defaults to "smtp") |
| Addr | Address mask (defaults INADDR_ANY) |
| Family | Address family (defaults to INET) |
| InputMailFilters | List of input mail filters for the daemon |
| Listen | Size of listen queue (defaults to 10) |
| Modifier | Options (flags) for the daemon |
| SndBufSize | Size of TCP send buffer |
| RcvBufSize | Size of TCP receive buffer |
| children | maximum number of children per daemon, see **MaxDaemonChildren**. |
| DeliveryMode | Delivery mode per daemon, see **DeliveryMode**. |
| refuseLA | RefuseLA per daemon |
| delayLA | DelayLA per daemon |
| queueLA | QueueLA per daemon |

The *Name* key is used for error messages and logging. The *Addr*ess mask may be a numeric address in IPv4 dot notation or IPv6 colon notation or a network name. Note that if a network name is specified, only the first IP address returned for it will be used. This may cause indeterminate behavior for network names that resolve to multiple addresses. Therefore, use of an address is recommended. The *Family* key defaults to INET (IPv4). IPv6 users who wish to also accept IPv6 connections should add additional Family=inet6 **DaemonPortOptions** lines. The *InputMailFilters* key overrides the default list of input mail filters listed in the **InputMailFilters** option. If multiple input mail filters are required, they must be separated by semicolons (not commas). *Modifier* can be a sequence (without any delimiters) of the following characters:

| | |
|---|---|
| a | always require authentication |
| b | bind to interface through which mail has been received |
| c | perform hostname canonification (.cf) |
| f | require fully qualified hostname (.cf) |
| s | Run smtps (SMTP over SSL) instead of smtp |
| u | allow unqualified addresses (.cf) |
| A | disable AUTH (overrides 'a' modifier) |
| C | don't perform hostname canonification |
| E | disallow ETRN (see RFC 2476) |
| O | optional; if opening the socket fails ignore it |
| S | don't offer STARTTLS |

That is, one way to specify a message submission agent (MSA) that always requires authentication is:

O DaemonPortOptions=Name=MSA, Port=587, M=Ea

The modifiers that are marked with "(.cf)" have only effect in the standard configuration file, in which they are available via **${daemon_flags}**. Notice: Do **not** use the ‘‘a’’ modifier on a public accessible MTA! It should only be used for a MSA that is accessed by authorized users for initial mail submission. Users must authenticate to use a MSA which has this option turned on. The flags ‘‘c’’ and ‘‘C’’ can change the default for hostname canonification in the *sendmail.cf* file. See the relevant documentation for FEATURE(nocanonify). The modifier ‘‘f’’ disallows addresses of the form **user@host** unless they are submitted directly. The flag ‘‘u’’ allows unqualified sender addresses, i.e., those without @host. ‘‘b’’ forces sendmail to bind to the interface through which the e-mail has been received for the outgoing connection. **WARNING:** Use ‘‘b’’ only if outgoing mail can be routed through the incoming connection's interface to its destination.

No attempt is made to catch problems due to a misconfiguration of this parameter, use it only for virtual hosting where each virtual interface can connect to every possible location. This will also override possible settings via **ClientPortOptions.** Note, *sendmail* will listen on a new socket for each occurence of the **DaemonPortOptions** option in a configuration file. The modifier ‘‘O’’ causes sendmail to ignore a socket if it can’t be opened. This applies to failures from the socket(2) and bind(2) calls.

DefaultAuthInfo [no short name] Filename that contains default authentication information for outgoing connections. This file must contain the user id, the authorization id, the password (plain text), the realm and the list of mechanisms to use on separate lines and must be readable by root (or the trusted user) only. If no realm is specified, **$j** is used. If no mechanisms are specified, the list given by **AuthMechanisms** is used. Notice: this option is deprecated and will be removed in future versions. Moreover, it doesn’t work for the MSP since it can’t read the file (the file must not be group/world-readable otherwise *sendmail* will complain). Use the authinfo ruleset instead which provides more control over the usage of the data anyway.

DefaultCharSet=*charset*
[no short name] When a message that has 8-bit characters but is not in MIME format is converted to MIME (see the EightBitMode option) a character set must be included in the Content-Type: header. This character set is normally set from the Charset= field of the mailer descriptor. If that is not set, the value of this option is used. If this option is not set, the value “unknown-8bit” is used.

DataFileBufferSize=*threshold*
[no short name] Set the *threshold*, in bytes, before a memory-based queue data file becomes disk-based. The default is 4096 bytes.

DeadLetterDrop=*file*
[no short name] Defines the location of the system-wide dead.letter file, formerly hardcoded to /usr/tmp/dead.letter. If this option is not set (the default), sendmail will not attempt to save to a system-wide dead.letter file in the event it cannot bounce the mail to the user or postmaster. Instead, it will rename the qf file as it has in the past when the dead.letter file could not be opened.

DefaultUser=*user:group*
[u] Set the default userid for mailers to *user:group*. If *group* is omitted and *user* is a user name (as opposed to a numeric user id) the default group listed in the /etc/passwd file for that user is used as the default group. Both *user* and *group* may be numeric. Mailers without the *S* flag in the mailer definition will run as this user. Defaults to 1:1. The value can also be given as a symbolic user name.[19]

DelayLA=*LA* [no short name] When the system load average exceeds *LA*, *sendmail* will sleep for one second on most SMTP commands and before accepting connections.

DeliverByMin=*time*
[0] Set minimum time for Deliver By SMTP Service Extension (RFC 2852). If 0, no time is listed, if less than 0, the extension is not offered, if greater than 0, it is listed as minimum time for the EHLO keyword DELIVERBY.

DeliveryMode=*x* [d] Deliver in mode *x*. Legal modes are:

[19]The old **g** option has been combined into the **DefaultUser** option.

i Deliver interactively (synchronously)
b Deliver in background (asynchronously)
q Just queue the message (deliver during queue run)
d Defer delivery and all map lookups (deliver during queue run)

Defaults to ‘‘b’’ if no option is specified, ‘‘i’’ if it is specified but given no argument (i.e., ‘‘Od’’ is equivalent to ‘‘Odi’’). The **−v** command line flag sets this to **i**. Note: for internal reasons, ‘‘i’’ does not work if a milter is enabled which can reject or delete recipients. In that case the mode will be changed to ‘‘b’’.

DialDelay=*sleeptime*
[no short name] Dial-on-demand network connections can see timeouts if a connection is opened before the call is set up. If this is set to an interval and a connection times out on the first connection being attempted *sendmail* will sleep for this amount of time and try again. This should give your system time to establish the connection to your service provider. Units default to seconds, so “DialDelay=5” uses a five second delay. Defaults to zero (no retry). This delay only applies to mailers which have the Z flag set.

DirectSubmissionModifiers=*modifiers*
Defines **${daemon_flags}** for direct (command line) submissions. If not set, **${daemon_flags}** is either "CC f" if the option **−G** is used or "c u" otherwise. Note that only the the "CC", "c", "f", and "u" flags are checked.

DontBlameSendmail=*option,option,...*
[no short name] In order to avoid possible cracking attempts caused by world- and group-writable files and directories, *sendmail* does paranoid checking when opening most of its support files. If for some reason you absolutely must run with, for example, a group-writable */etc* directory, then you will have to turn off this checking (at the cost of making your system more vulnerable to attack). The possible arguments have been described earlier. The details of these flags are described above. **Use of this option is not recommended.**

DontExpandCnames
[no short name] The standards say that all host addresses used in a mail message must be fully canonical. For example, if your host is named “Cruft.Foo.ORG” and also has an alias of “FTP.Foo.ORG”, the former name must be used at all times. This is enforced during host name canonification ($[ ... $] lookups). If this option is set, the protocols are ignored and the “wrong” thing is done. However, the IETF is moving toward changing this standard, so the behavior may become acceptable. Please note that hosts downstream may still rewrite the address to be the true canonical name however.

DontInitGroups [no short name] If set, *sendmail* will avoid using the initgroups(3) call. If you are running NIS, this causes a sequential scan of the groups.byname map, which can cause your NIS server to be badly overloaded in a large domain. The cost of this is that the only group found for users will be their primary group (the one in the password file), which will make file access permissions somewhat more restrictive. Has no effect on systems that don’t have group lists.

DontProbeInterfaces
[no short name] *Sendmail* normally finds the names of all interfaces active on your machine when it starts up and adds their name to the **$=w** class of known host aliases. If you have a large number of virtual interfaces or if your DNS inverse lookups are slow this can be time consuming. This option turns off that probing. However, you will need to be certain to include all variant names in the **$=w** class by some other mechanism. If set to **loopback**, loopback interfaces (e.g., lo0) will not be probed.

DontPruneRoutes [R] Normally, *sendmail* tries to eliminate any unnecessary explicit routes when sending an error message (as discussed in RFC 1123 § 5.2.6). For example, when sending an error message to

<@known1,@known2,@known3:user@unknown>

*sendmail* will strip off the "@known1,@known2" 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.

DoubleBounceAddress=*error-address*

[no short name] If an error occurs when sending an error message, send the error report (termed a "double bounce" because it is an error "bounce" that occurs when trying to send another error "bounce") to the indicated address. The address is macro expanded at the time of delivery. If not set, defaults to "postmaster". If set to an empty string, double bounces are dropped.

EightBitMode=*action*

[8] Set handling of eight-bit data. There are two kinds of eight-bit data: that declared as such using the **BODY=8BITMIME** ESMTP declaration or the **−B8BITMIME** command line flag, and undeclared 8-bit data, that is, input that just happens to be eight bits. There are three basic operations that can happen: undeclared 8-bit data can be automatically converted to 8BITMIME, undeclared 8-bit data can be passed as-is without conversion to MIME (‘‘just send 8’’), and declared 8-bit data can be converted to 7-bits for transmission to a non-8BITMIME mailer. The possible *action*s are:

s Reject undeclared 8-bit data (‘‘strict’’)
m Convert undeclared 8-bit data to MIME (‘‘mime’’)
p Pass undeclared 8-bit data (‘‘pass’’)

In all cases properly declared 8BITMIME data will be converted to 7BIT as needed.

ErrorHeader=*file-or-message*

[E] 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.

ErrorMode=*x* [e] 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 (when applicable) and give zero exit stat always

Note that the last mode, "e", is for Berknet error processing and should not be used in normal circumstances. Note, too, that mode "q", only applies to errors recognized before sendmail forks for background delivery.

FallbackMXhost=*fallbackhost*

[V] If specified, the *fallbackhost* acts like a very low priority MX on every host. MX records will be looked up for this host, unless the name is surrounded by square brackets. This is intended to be used by sites with poor network connectivity. Messages which are undeliverable due to temporary address failures (e.g., DNS failure) also go to the FallbackMXhost.

FallBackSmartHost=*hostname*
: If specified, the *FallBackSmartHost* will be used in a last-ditch effort for each host. This is intended to be used by sites with "fake internal DNS", e.g., a company whose DNS accurately reflects the world inside that company's domain but not outside.

FastSplit
: [no short name] If set to a value greater than zero (the default is one), it suppresses the MX lookups on addresses when they are initially sorted, i.e., for the first delivery attempt. This usually results in faster envelope splitting unless the MX records are readily available in a local DNS cache. To enforce initial sorting based on MX records set **FastSplit** to zero. If the mail is submitted directly from the command line, then the value also limits the number of processes to deliver the envelopes; if more envelopes are created they are only queued up and must be taken care of by a queue run. Since the default submission method is via SMTP (either from a MUA or via the MSP), the value of **FastSplit** is seldom used to limit the number of processes to deliver the envelopes.

ForkEachJob
: [Y] If set, deliver each job that is run from the queue in a separate process.

ForwardPath=*path*
: [J] Set the path for searching for users' .forward files. The default is "$z/.forward". Some sites that use the automounter may prefer to change this to "/var/forward/$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, "/var/forward/$u:$z/.forward" will search first in /var/forward/*username* and then in ~*username*/.forward (but only if the first file does not exist).

HeloName=*name*
: [no short name] Set the name to be used for HELO/EHLO (instead of $j).

HoldExpensive
: [c] If an outgoing mailer is marked as being expensive, don't connect immediately.

HostsFile=*path*
: [no short name] The path to the hosts database, normally "/etc/hosts". This option is only consulted when sendmail is canonifying addresses, and then only when "files" is in the "hosts" service switch entry. In particular, this file is *never* used when looking up host addresses; that is under the control of the system *gethostbyname*(3) routine.

HostStatusDirectory=*path*
: [no short name] The location of the long term host status information. When set, information about the status of hosts (e.g., host down or not accepting connections) will be shared between all *sendmail* processes; normally, this information is only held within a single queue run. This option requires a connection cache of at least 1 to function. If the option begins with a leading '/', it is an absolute pathname; otherwise, it is relative to the mail queue directory. A suggested value for sites desiring persistent host status is ".hoststat" (i.e., a subdirectory of the queue directory).

IgnoreDots
: [i] Ignore dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

InputMailFilters=*name,name,...*
: A comma separated list of filters which determines which filters (see the "X — Mail Filter (Milter) Definitions" section) and the invocation sequence are contacted for incoming SMTP messages. If none are set, no filters will be contacted.

LDAPDefaultSpec=*spec*
: [no short name] Sets a default map specification for LDAP maps. The value should only contain LDAP specific settings such as "-h host -p port -d bindDN". The settings will be used for all LDAP maps unless the individual map specification overrides a setting. This option should be set before any LDAP maps are defined.

LogLevel=*n*
: [L] Set the log level to *n*. Defaults to 9.

M*x value*
: [no long version] Set the macro *x* to *value*. This is intended only for use from the command line. The **−M** flag is preferred.

MailboxDatabase
: [no short name] Type of lookup to find information about local mailboxes, defaults to ''pw'' which uses *getpwnam*. Other types can be introduced by adding them to the source code, see libsm/mbdb.c for details.

UseMSP
: [no short name] Use as mail submission program, i.e., allow group writable queue files if the group is the same as that of a set-group-ID sendmail binary. See the file **sendmail/SECURITY** in the distribution tarball.

MatchGECOS
: [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.

MaxAliasRecursion=*N*
: [no short name] The maximum depth of alias recursion (default: 10).

MaxDaemonChildren=*N*
: [no short name] If set, *sendmail* will refuse connections when it has more than *N* children processing incoming mail or automatic queue runs. This does not limit the number of outgoing connections. If the default **DeliveryMode** (background) is used, then *sendmail* may create an almost unlimited number of children (depending on the number of transactions and the relative execution times of mail receiption and mail delivery). If the limit should be enforced, then a **DeliveryMode** other than background must be used. If not set, there is no limit to the number of children -- that is, the system load average controls this.

MaxHeadersLength=*N*
: [no short name] The maximum length of the sum of all headers. This can be used to prevent a denial of service attack. The default is no limit.

MaxHopCount=*N*
: [h] 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.

MaxMessageSize=*N*
: [no short name] Specify the maximum message size to be advertised in the ESMTP EHLO response. Messages larger than this will be rejected. If set to a value greater than zero, that value will be listed in the SIZE response, otherwise SIZE is advertised in the ESMTP EHLO response without a parameter.

MaxMimeHeaderLength=*N[/M]*
: [no short name] Sets the maximum length of certain MIME header field values to *N* characters. These MIME header fields are determined by being a member of class {checkMIMETextHeaders}, which currently contains only the header Content-Description. For some of these headers which take parameters, the maximum length of each parameter is set to *M* if specified. If */M* is not specified, one half of *N* will be used. By default, these values are 2048 and 1024, respectively. To

allow any length, a value of 0 can be specified.

MaxNOOPCommands=*N*
: Override the default of **MAXNOOPCOMMANDS** for the number of *useless* commands, see Section "Measures against Denial of Service Attacks".

MaxQueueChildren=*N*
: [no short name] When set, this limits the number of concurrent queue runner processes to *N*. This helps to control the amount of system resources used when processing the queue. When there are multiple queue groups defined and the total number of queue runners for these queue groups would exceed *MaxQueueChildren* then the queue groups will not all run concurrently. That is, some portion of the queue groups will run concurrently such that *MaxQueueChildren* will not be exceeded, while the remaining queue groups will be run later (in round robin order). See also *MaxRunnersPerQueue* and the section **Queue Group Declaration**. Notice: *sendmail* does not count individual queue runners, but only sets of processes that act on a workgroup. Hence the actual number of queue runners may be lower than the limit imposed by *MaxQueueChildren*. This discrepancy can be large if some queue runners have to wait for a slow server and if short intervals are used.

MaxQueueRunSize=*N*
: [no short name] The maximum number of jobs that will be processed in a single queue run. If not set, there is no limit on the size. If you have very large queues or a very short queue run interval this could be unstable. However, since the first *N* jobs in queue directory order are run (rather than the *N* highest priority jobs) this should be set as high as possible to avoid “losing” jobs that happen to fall late in the queue directory. Note: this option also restricts the number of entries printed by *mailq*. That is, if *MaxQueueRunSize* is set to a value **N** larger than zero, then only **N** entries are printed per queue group.

MaxRecipientsPerMessage=*N*
: [no short name] The maximum number of recipients that will be accepted per message in an SMTP transaction. Note: setting this too low can interfere with sending mail from MUAs that use SMTP for initial submission. If not set, there is no limit on the number of recipients per envelope.

MaxRunnersPerQueue=*N*
: [no short name] This sets the default maximum number of queue runners for queue groups. Up to *N* queue runners will work in parallel on a queue group’s messages. This is useful where the processing of a message in the queue might delay the processing of subsequent messages. Such a delay may be the result of non-erroneous situations such as a low bandwidth connection. May be overridden on a per queue group basis by setting the *Runners* option; see the section **Queue Group Declaration**. The default is 1 when not set.

MeToo
: [m] Send to me too, even if I am in an alias expansion. This option is deprecated and will be removed from a future version.

Milter
: [no short name] This option has several sub(sub)options. The names of the suboptions are separated by dots. At the first level the following options are available:

| | |
|---|---|
| LogLevel | Log level for input mail filter actions, defaults to LogLevel. |
| macros | Specifies list of macro to transmit to filters.<br>See list below. |

The ‘‘macros’’ option has the following suboptions which specify the list of macro to transmit to milters after a certain event occurred.

| | |
|---|---|
| connect | After session connection start |
| helo | After EHLO/HELO command |
| envfrom | After MAIL From command |
| envrcpt | After RCPT To command |
| data | After DATA command. |
| eoh | After DATA command and header |
| eom | After DATA command and terminating ''.'' |

By default the lists of macros are empty. Example:

O Milter.LogLevel=12
O Milter.macros.connect=j, _, {daemon_name}

MinFreeBlocks=*N*
: [b] 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.

MinQueueAge=*age*
: [no short name] Don't process any queued jobs that have been in the queue less than the indicated time interval. This is intended to allow you to get responsiveness by processing the queue fairly frequently without thrashing your system by trying jobs too often. The default units are minutes. Note: This option is ignored for queue runs that select a subset of the queue, i.e., "−q[!][I|R|S|Q][string]"

MustQuoteChars=*s*
: [no short name] Sets the list of characters that must be quoted if used in a full name that is in the phrase part of a ''phrase <address>'' syntax. The default is ''´.''. The characters ''@,;:\()[]'' are always added to this list.

NiceQueueRun
: [no short name] The priority of queue runners (nice(3)). This value must be greater or equal zero.

NoRecipientAction
: [no short name] The action to take when you receive a message that has no valid recipient headers (To:, Cc:, Bcc:, or Apparently-To: — the last included for back compatibility with old *sendmail*s). It can be **None** to pass the message on unmodified, which violates the protocol, **Add-To** to add a To: header with any recipients it can find in the envelope (which might expose Bcc: recipients), **Add-Apparently-To** to add an Apparently-To: header (this is only for back-compatibility and is officially deprecated), **Add-To-Undisclosed** to add a header "To: undisclosed-recipients:;" to make the header legal without disclosing anything, or **Add-Bcc** to add an empty Bcc: header.

OldStyleHeaders
: [o] Assume that the headers may be in old format, i.e., spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names. Defaults to off.

OperatorChars=*charlist*
: [$o macro] The list of characters that are considered to be "operators", that is, characters that delimit tokens. All operator characters are tokens by themselves; sequences of non-operator characters are also tokens. White space characters separate tokens but are not tokens themselves — for example, "AAA.BBB" has three tokens, but "AAA BBB" has two. If not set, OperatorChars defaults to ". : @ [ ]"; additionally, the characters "( ) < > , ;" are always operators. Note that

OperatorChars must be set in the configuration file before any rulesets.

PidFile=*filename* [no short name] Filename of the pid file. (default is _PATH_SENDMAILPID). The *filename* is macro-expanded before it is opened, and unlinked when *sendmail* exits.

PostmasterCopy=*postmaster*
[P] If set, copies of error messages will be sent to the named *postmaster*. Only the header of the failed message is sent. Errors resulting from messages with a negative precedence will not be sent. Since most errors are user problems, this is probably not a good idea on large sites, and arguably contains all sorts of privacy violations, but it seems to be popular with certain operating systems vendors. The address is macro expanded at the time of delivery. Defaults to no postmaster copies.

PrivacyOptions= *opt,opt,...*
[p] 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 *opt*ions 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, implies noverb. |
| needvrfyhelo | Insist on HELO or EHLO command before VRFY |
| novrfy | Disallow VRFY entirely |
| noetrn | Disallow ETRN entirely |
| noverb | Disallow VERB entirely |
| restrictmailq | Restrict mailq command |
| restrictqrun | Restrict −q command line flag |
| restrictexpand | Restrict −bv and −v command line flags |
| noreceipts | Don't return success DSNs[20] |
| nobodyreturn | Don't return the body of a message with DSNs |
| goaway | Disallow essentially all SMTP status queries |
| authwarnings | Put X-Authentication-Warning: headers in messages and log warnings |
| noactualrecipient | Don't put X-Actual-Recipient lines in DSNs which reveal the actual account that addresses map to. |

The "goaway" pseudo-flag sets all flags except "noreceipts", "restrictmailq", "restrictqrun", "restrictexpand", "noetrn", and "nobodyreturn". If mailq is restricted, 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. The "restrictexpand" pseudo-flag instructs *sendmail* to drop privileges when the **−bv** option is given by users who are neither root nor the TrustedUser so users cannot read private aliases, forwards, or :include: files. It will add the "NonRootSafeAddr" to the "DontBlameSendmail" option to prevent misleading unsafe address warnings. It also overrides the **−v** (verbose) command line option to prevent information leakage. Authentication Warnings add warnings about various conditions that may indicate attempts to spoof the mail system, such as using a non-standard queue directory.

[20]N.B.: the **noreceipts** flag turns off support for RFC 1891 (Delivery Status Notification).

ProcessTitlePrefix=*string*
[no short name] Prefix the process title shown on 'ps' listings with *string*. The *string* will be macro processed.

QueueDirectory=*dir*
[Q] The QueueDirectory option serves two purposes. First, it specifies the directory or set of directories that comprise the default queue group. Second, it specifies the directory D which is the ancestor of all queue directories, and which sendmail uses as its current working directory. When sendmail dumps core, it leaves its core files in D. There are two cases. If *dir* ends with an asterisk (eg, */var/spool/mqueue/qd**), then all of the directories or symbolic links to directories beginning with 'qd' in */var/spool/mqueue* will be used as queue directories of the default queue group, and */var/spool/mqueue* will be used as the working directory D. Otherwise, *dir* must name a directory (usually */var/spool/mqueue*): the default queue group consists of the single queue directory *dir*, and the working directory D is set to *dir*. To define additional groups of queue directories, use the configuration file 'Q' command. Do not change the queue directory structure while sendmail is running.

QueueFactor=*factor*
[q] Use *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (**QueueLA** option) to determine the maximum message priority that will be sent. Defaults to 600000.

QueueLA=*LA*
[x] When the system load average exceeds *LA* and the **QueueFactor** (**q**) option divided by the difference in the current load average and the **QueueLA** option plus one is less than the priority of the message, just queue messages (i.e., don't try to send them). Defaults to 8 multiplied by the number of processors online on the system (if that can be determined).

QueueFileMode=*mode*
[no short name] Default permissions for queue files (octal). If not set, sendmail uses 0600 unless its real and effective uid are different in which case it uses 0644.

QueueSortOrder=*algorithm*
[no short name] Sets the *algorithm* used for sorting the queue. Only the first character of the value is used. Legal values are "host" (to order by the name of the first host name of the first recipient), "filename" (to order by the name of the queue file name), "time" (to order by the submission/creation time), "random" (to order randomly), "modification" (to order by the modification time of the qf file (older entries first)), "none" (to not order), and "priority" (to order by message priority). Host ordering makes better use of the connection cache, but may tend to process low priority messages that go to a single host over high priority messages that go to several hosts; it probably shouldn't be used on slow network links. Filename and modification time ordering saves the overhead of reading all of the queued items before starting the queue run. Creation (submission) time ordering is almost always a bad idea, since it allows large, bulk mail to go out before smaller, personal mail, but may have applicability on some hosts with very fast connections. Random is useful if several queue runners are started by hand which try to drain the same queue since odds are they will be working on different parts of the queue at the same time. Priority ordering is the default.

QueueTimeout=*timeout*
[T] A synonym for "Timeout.queuereturn". Use that form instead of the "QueueTimeout" form.

RandFile [no short name] Name of file containing random data or the name of the UNIX socket if EGD is used. A (required) prefix "egd:" or "file:" specifies the type. STARTTLS requires this filename if the compile flag HASURANDOMDEV is not set (see sendmail/README).

ResolverOptions=*options*
: [I] Set resolver options. Values can be set using **+***flag* and cleared using **−***flag*; the *flag*s can be "debug", "aaonly", "usevc", "primary", "igntc", "recurse", "defnames", "stayopen", "use_inet6", or "dnsrch". The string "HasWildcardMX" (without a **+** or **−**) can be specified to turn off matching against MX records when doing name canonifications. The string "WorkAroundBrokenAAAA" (without a **+** or **−**) can be specified to work around some broken nameservers which return SERVFAIL (a temporary failure) on T_AAAA (IPv6) lookups. Notice: it might be necessary to apply the same (or similar) options to *submit.cf* too.

RequiresDirfsync [no short name] This option can be used to override the compile time flag **REQUIRES_DIR_FSYNC** at runtime by setting it to false. If the compile time flag is not set, the option is ignored. The flag turns on support for file systems that require to call *fsync()* for a directory if the meta-data in it has been changed. This should be turned on at least for older versions of ReiserFS; it is enabled by default for Linux. According to some information this flag is not needed anymore for kernel 2.4.16 and newer.

RrtImpliesDsn [R] If this option is set, a "Return-Receipt-To:" header causes the request of a DSN, which is sent to the envelope sender as required by RFC 1891, not to the address given in the header.

RunAsUser=*user* [no short name] The *user* parameter may be a user name (looked up in */etc/passwd*) or a numeric user id; either form can have ":group" attached (where group can be numeric or symbolic). If set to a non-zero (non-root) value, *sendmail* will change to this user id shortly after startup[21]. This avoids a certain class of security problems. However, this means that all ".forward" and ":include:" files must be readable by the indicated *user* and all files to be written must be writable by *user* Also, all file and program deliveries will be marked unsafe unless the option **DontBlameSendmail=NonRootSafeAddr** is set, in which case the delivery will be done as *user*. It is also incompatible with the **SafeFileEnvironment** option. In other words, it may not actually add much to security on an average system, and may in fact detract from security (because other file permissions must be loosened). However, it should be useful on firewalls and other places where users don't have accounts and the aliases file is well constrained.

RecipientFactor=*fact*
: [y] The indicated *fact*or is added to the priority (thus *lowering* the priority of the job) for each recipient, i.e., this value penalizes jobs with large numbers of recipients. Defaults to 30000.

RefuseLA=*LA* [X] When the system load average exceeds *LA*, refuse incoming SMTP connections. Defaults to 12 multiplied by the number of processors online on the system (if that can be determined).

RejectLogInterval=*timeout*
: [no short name] Log interval when refusing connections for this long (default: 3h).

[21]When running as a daemon, it changes to this user after accepting a connection but before reading any SMTP commands.

RetryFactor=*fact* [Z] The *fact*or is added to the priority every time a job is processed. Thus, each time a job is processed, its priority will be decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time. Defaults to 90000.

SafeFileEnvironment=*dir*
[no short name] If this option is set, *sendmail* will do a *chroot*(2) call into the indicated *dir*ectory before doing any file writes. If the file name specified by the user begins with *dir*, that partial path name will be stripped off before writing, so (for example) if the SafeFileEnvironment variable is set to "/safe" then aliases of "/safe/logs/file" and "/logs/file" actually indicate the same file. Additionally, if this option is set, *sendmail* refuses to deliver to symbolic links.

SaveFromLine [f] Save UNIX-style "From" lines at the front of headers. Normally they are assumed redundant and discarded.

SendMimeErrors [j] If set, send error messages in MIME format (see RFC 2045 and RFC 1344 for details). If disabled, *sendmail* will not return the DSN keyword in response to an EHLO and will not do Delivery Status Notification processing as described in RFC 1891.

ServerCertFile [no short name] File containing the certificate of the server, i.e., this certificate is used when sendmail acts as server (used for STARTTLS).

ServerKeyFile [no short name] File containing the private key belonging to the server certificate (used for STARTTLS).

ServiceSwitchFile=*filename*
[no short name] If your host operating system has a service switch abstraction (e.g., /etc/nsswitch.conf on Solaris or /etc/svc.conf on Ultrix and DEC OSF/1) that service will be consulted and this option is ignored. Otherwise, this is the name of a file that provides the list of methods used to implement particular services. The syntax is a series of lines, each of which is a sequence of words. The first word is the service name, and following words are service types. The services that *sendmail* consults directly are "aliases" and "hosts." Service types can be "dns", "nis", "nisplus", or "files" (with the caveat that the appropriate support must be compiled in before the service can be referenced). If ServiceSwitchFile is not specified, it defaults to /etc/mail/service.switch. If that file does not exist, the default switch is:

```
aliases     files
hosts       dns nis files
```

The default file is "/etc/mail/service.switch".

SevenBitInput [7] Strip input to seven bits for compatibility with old systems. This shouldn't be necessary.

SharedMemoryKey
[no short name] Key to use for shared memory segment; if not set (or 0), shared memory will not be used. If set to -1 *sendmail* can select a key itself provided that also **SharedMemoryKeyFile** is set. Requires support for shared memory to be compiled into *sendmail*. If this option is set, *sendmail* can share some data between different instances. For example, the number of entries in a queue directory or the available space in a file system. This allows for more efficient program execution, since only one process needs to update the data instead of each individual process gathering the data each time it is required.

SharedMemoryKeyFile
[no short name] If **SharedMemoryKey** is set to -1 then the automatically selected

shared memory key will be stored in the specified file.

SingleLineFromHeader
: [no short name] If set, From: lines that have embedded newlines are unwrapped onto one line. This is to get around a botch in Lotus Notes that apparently cannot understand legally wrapped RFC 822 headers.

SingleThreadDelivery
: [no short name] If set, a client machine will never try to open two SMTP connections to a single server machine at the same time, even in different processes. That is, if another *sendmail* is already talking to some host a new *sendmail* will not open another connection. This property is of mixed value; although this reduces the load on the other machine, it can cause mail to be delayed (for example, if one *sendmail* is delivering a huge message, other *sendmail*s won't be able to send even small messages). Also, it requires another file descriptor (for the lock file) per connection, so you may have to reduce the **ConnectionCacheSize** option to avoid running out of per-process file descriptors. Requires the **HostStatusDirectory** option.

SmtpGreetingMessage=*message*
: [$e macro] The message printed when the SMTP server starts up. Defaults to "$j Sendmail $v ready at $b".

SoftBounce
: If set, issue temporary errors (4xy) instead of permanent errors (5xy). This can be useful during testing of a new configuration to avoid erroneous bouncing of mails.

StatusFile=*file*
: [S] Log summary statistics in the named *file*. If no file name is specified, "statistics" is used. If not set, no summary statistics are saved. This file does not grow in size. It can be printed using the *mailstats*(8) program.

SuperSafe
: [s] This option can be set to True, False, Interactive, or PostMilter. If set to True, *sendmail* will be super-safe when running things, i.e., always instantiate the queue file, even if you are going to attempt immediate delivery. *Sendmail* always instantiates the queue file before returning control to the client under any circumstances. This should really *always* be set to True. The Interactive value has been introduced in 8.12 and can be used together with **DeliveryMode=i**. It skips some synchronization calls which are effectively doubled in the code execution path for this mode. If set to PostMilter, *sendmail* defers synchronizing the queue file until any milters have signaled acceptance of the message. PostMilter is useful only when *sendmail* is running as an SMTP server; in all other situations it acts the same as True.

TLSSrvOptions
: [no short name] List of options for SMTP STARTTLS for the server consisting of single characters with intervening white space or commas. The flag ''V'' disables client verification, and hence it is not possible to use a client certificate for relaying. Currently there are no other flags available.

TempFileMode=*mode*
: [F] The file mode for transcript files, files to which *sendmail* delivers directly, files in the **HostStatusDirectory**, and **StatusFile**. It is interpreted in octal by default. Defaults to 0600.

Timeout.*type*= *timeout*
: [r; subsumes old T option as well] Set timeout values. For more information, see section 4.1.

TimeZoneSpec=*tzinfo*
: [t] 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.

TrustedUser=*user* [no short name] The *user* parameter may be a user name (looked up in */etc/passwd*) or a numeric user id. Trusted user for file ownership and starting the daemon. If set, generated alias databases and the control socket (if configured) will automatically be owned by this user.

TryNullMXList [w] If this system is the "best" (that is, lowest preference) MX for a given host, its configuration rules should normally detect this situation and treat that condition specially by forwarding the mail to a UUCP feed, treating it as local, or whatever. 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 disrecommended.

UnixFromLine=*fromline*
[$l macro] Defines the format used when *sendmail* must add a UNIX-style From_ line (that is, a line beginning "From<space>user"). Defaults to "From $g  $d". Don't change this unless your system uses a different UNIX mailbox format (very unlikely).

UnsafeGroupWrites
[no short name] If set (default), :include: and .forward files that are group writable are considered "unsafe", that is, they cannot reference programs or write directly to files. World writable :include: and .forward files are always unsafe. Note: use **DontBlameSendmail** instead; this option is deprecated.

UseErrorsTo [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. This option is disrecommended and deprecated.

UserDatabaseSpec=*udbspec*
[U] The user database specification.

Verbose [v] Run in verbose mode. If this is set, *sendmail* adjusts options **HoldExpensive** (old **c**) and **DeliveryMode** (old **d**) so that all mail is delivered completely in a single job so that you can see the entire delivery process. Option **Verbose** should *never* be set in the configuration file; it is intended for command line use only. Note that the use of option **Verbose** can cause authentication information to leak, if you use a sendmail client to authenticate to a server. If the authentication mechanism uses plain text passwords (as with LOGIN or PLAIN), then the password could be compromised. To avoid this, do not install sendmail set-user-ID root, and disable the **VERB** SMTP command with a suitable **PrivacyOptions** setting.

XscriptFileBufferSize=*threshold*
[no short name] Set the *threshold*, in bytes, before a memory-based queue transcript file becomes disk-based. The default is 4096 bytes.

All options can be specified on the command line using the −O or −o flag, but most will cause *sendmail* to relinquish its set-user-ID permissions. The options that will not cause this are SevenBitInput [7], EightBitMode [8], MinFreeBlocks [b], CheckpointInterval [C], DeliveryMode [d], ErrorMode [e], IgnoreDots [i], SendMimeErrors [j], LogLevel [L], MeToo [m], OldStyleHeaders [o], PrivacyOptions [p], SuperSafe [s], Verbose [v], QueueSortOrder, MinQueueAge, DefaultCharSet, Dial Delay, NoRecipientAction, ColonOkInAddr, MaxQueueRunSize, SingleLineFromHeader, and AllowBogusHELO. Actually, PrivacyOptions [p] given on the command line are added to those already specified in the *sendmail.cf* file, i.e., they can't be reset. Also, M (define macro) when defining the r or s macros is also considered "safe".

### 5.7. 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 less than 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. For example, our 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) didn't 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*.

### 5.8. 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.

**N.B.:** these version *levels* have nothing to do with the version *number* on the files. For example, as of this writing version 10 config files (specifically, 8.10) used version level 9 configurations.

"Old" configuration files are defined as version level one. Version level two files make the following changes:

(1) Host name canonification ($[ ... $]) appends a dot if the name is recognized; this gives the config 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.)

(2) 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.

(3) 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.

Version level six configuration files change many of the local processing options (such as aliasing and matching the beginning of the address for '|' characters) to be mailer flags; this allows fine-grained control over the special local processing. Level six configuration files may also use long option names. The **ColonOkInAddr** option (to allow colons in the local-part of addresses) defaults **on** for lower numbered configuration files; the configuration file requires some additional intelligence to properly handle the RFC 822 group construct.

Version level seven configuration files used new option names to replace old macros (**$e** became **SmtpGreetingMessage**, **$l** became **UnixFromLine**, and **$o** became **OperatorChars**. Also, prior to version seven, the **F=q** flag (use 250 instead of 252 return value for SMTP VRFY commands) was assumed.

Version level eight configuration files allow **$#** on the left hand side of ruleset lines.

Version level nine configuration files allow parentheses in rulesets, i.e. they are not treated as comments and hence removed.

Version level ten configuration files allow queue group definitions.

The **V** line may have an optional **/***vendor* to indicate that this configuration file uses modifications specific to a particular vendor[22]. You may use "/Berkeley" to emphasize that this configuration file uses the Berkeley dialect of *sendmail*.

### 5.9. 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 in to *sendmail*. 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.

The *arguments* are passed to the map for arbitrary use. Most map classes can interpolate these arguments into their values using the syntax "%*n*" (where *n* is a digit) to indicate the corresponding *argument*. Argument "%0" indicates the database key. For example, the rule

```
R$− ! $+        $: $(uucp $1 $@ $2 $: $2 @ $1 . UUCP $)
```

Looks up the UUCP name in a (user defined) UUCP map; if not found it turns it into ".UUCP" form. The database might contain records like:

```
decvax          %1@%0.DEC.COM
research        %1@%0.ATT.COM
```

Note that *default* clauses never do this mapping.

---

[22] And of course, vendors are encouraged to add themselves to the list of recognized vendors by editing the routine *setvendor* in *conf.c*. Please send e-mail to sendmail@Sendmail.ORG to register your vendor dialect.

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 many defined classes.

dbm — Database lookups using the ndbm(3) library. *Sendmail* must be compiled with **NDBM** defined.

btree — Database lookups using the btree interface to the Berkeley DB library. *Sendmail* must be compiled with **NEWDB** defined.

hash — Database lookups using the hash interface to the Berkeley DB library. *Sendmail* must be compiled with **NEWDB** defined.

nis — NIS lookups. *Sendmail* must be compiled with **NIS** defined.

nisplus — NIS+ lookups. *Sendmail* must be compiled with **NISPLUS** defined. The argument is the name of the table to use for lookups, and the **−k** and **−v** flags may be used to set the key and value columns respectively.

hesiod — Hesiod lookups. *Sendmail* must be compiled with **HESIOD** defined.

ldap — LDAP X500 directory lookups. *Sendmail* must be compiled with **LDAPMAP** defined. The map supports most of the standard arguments and most of the command line arguments of the *ldapsearch* program. Note that, by default, if a single query matches multiple values, only the first value will be returned unless the **−z** (value separator) map flag is set. Also, the **−1** map flag will treat a multiple value return as if there were no matches.

netinfo — NeXT NetInfo lookups. *Sendmail* must be compiled with **NETINFO** defined.

text — Text file lookups. The format of the text file is defined by the **−k** (key field number), **−v** (value field number), and **−z** (field delimiter) flags.

ph — PH query map. Contributed and supported by Mark Roth, roth@uiuc.edu. For more information, consult the web site "http://www-dev.cites.uiuc.edu/sendmail/".

nsd — nsd map for IRIX 6.5 and later. Contributed and supported by Bob Mende of SGI, mende@sgi.com.

stab — Internal symbol table lookups. Used internally for aliasing.

implicit — Really should be called "alias" — this is used to get the default lookups for alias files, and is the default if no class is specified for alias files.

user — Looks up users using *getpwnam*(3). The **−v** flag can be used to specify the name of the field to return (although this is normally used only to check the existence of a user).

host — Canonifies host domain names. Given a host name it calls the name server to find the canonical name for that host.

bestmx — Returns the best MX record for a host name given as the key. The current machine is always preferred — that is, if the current machine is one of the hosts listed as a lowest-preference MX record, then it will be guaranteed to be returned. This can be used to find out if this machine is the target for an MX record, and mail can be accepted on that basis. If the **−z** flag is given, then all MX names are returned, separated by the given delimiter.

dns
: This map requires the option -R to specify the DNS resource record type to lookup. The following types are supported: A, AAAA, AFSDB, CNAME, MX, NS, PTR, SRV, and TXT. A map lookup will return only one record. Hence for some types, e.g., MX records, the return value might be a random element of the list due to randomizing in the DNS resolver.

sequence
: The arguments on the ‘K’ line are a list of maps; the resulting map searches the argument maps in order until it finds a match for the indicated key. For example, if the key definition is:

```
Kmap1 ...
Kmap2 ...
Kseqmap sequence map1 map2
```

then a lookup against “seqmap” first does a lookup in map1. If that is found, it returns immediately. Otherwise, the same key is used for map2.

syslog
: the key is logged via *syslogd* (8). The lookup returns the empty string.

switch
: Much like the “sequence” map except that the order of maps is determined by the service switch. The argument is the name of the service to be looked up; the values from the service switch are appended to the map name to create new map names. For example, consider the key definition:

```
Kali switch aliases
```

together with the service switch entry:

```
aliases            nis files
```

This causes a query against the map “ali” to search maps named “ali.nis” and “ali.files” in that order.

dequote
: Strip double quotes (") from a name. It does not strip backslashes, and 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.

regex
: The map definition on the **K** line contains a regular expression. Any key input is compared to that expression using the POSIX regular expressions routines regcomp(), regerr(), and regexec(). Refer to the documentation for those routines for more information about the regular expression matching. No rewriting of the key is done if the **−m** flag is used. Without it, the key is discarded or if **−s** if used, it is substituted by the substring matches, delimited by **$|** or the string specified with the the **−d** flag. The flags available for the map are

-n not
-f case sensitive
-b basic regular expressions (default is extended)
-s substring match
-d set the delimiter used for -s
-a append string to key
-m match only, do not replace/discard value
-D perform no lookup in deferred delivery mode.

The **−s** flag can include an optional parameter which can be used to select the substrings in the result of the lookup. For example,

-s1,3,4

Notes: to match a **$** in a string, \$$ must be used. If the pattern contains spaces, they must be replaced with the blank substitution character, unless it is space itself.

program The arguments on the **K** line are the pathname to a program and any initial parameters to be passed. When the map is called, the key is added to the initial parameters and the program is invoked as the default user/group id. The first line of standard output is returned as the value of the lookup. This has many potential security problems, and has terrible performance; it should be used only when absolutely necessary.

macro Set or clear a macro value. To set a macro, pass the value as the first argument in the map lookup. To clear a macro, do not pass an argument in the map lookup. The map always returns the empty string. Example of typical usage include:

```
Kstorage macro

...

# set macro ${MyMacro} to the ruleset match
R$+ $: $(storage {MyMacro} $@ $1 $) $1
# set macro ${MyMacro} to an empty string
R$* $: $(storage {MyMacro} $@ $) $1
# clear macro ${MyMacro}
R$− $: $(storage {MyMacro} $) $1
```

arith Perform simple arithmetic operations. The operation is given as key, currently +, -, *, /, %, |, & (bitwise OR, AND), l (for less than), =, and r (for random) are supported. The two operands are given as arguments. The lookup returns the result of the computation, i.e., TRUE or FALSE for comparisons, integer values otherwise. The r operator returns a pseudo-random number whose value lies between the first and second operand (which requires that the first operand is smaller than the second). All options which are possible for maps are ignored. A simple example is:

```
Kcomp arith

...

Scheck_etrn
R$* $: $(comp l $@ $&{load_avg} $@ 7 $) $1
RFALSE$# error ...
```

socket

The socket map uses a simple request/reply protocol over TCP or UNIX domain sockets to query an external server. Both requests and replies are text based and encoded as netstrings, i.e., a string "hello there" becomes:

    11:hello there,

Note: neither requests nor replies end with CRLF.

The request consists of the database map name and the lookup key separated by a space character:

    <mapname> ’ ’ <key>

The server responds with a status indicator and the result (if any):

    <status> ’ ’ <result>

The status indicator specifies the result of the lookup operation itself and is one of the following upper case words:

| | |
|---|---|
| OK | the key was found, result contains the looked up value |
| NOTFOUND | the key was not found, the result is empty |
| TEMP | a temporary failure occured |
| TIMEOUT | a timeout occured on the server side |
| PERM | a permanent failure occured |

In case of errors (status TEMP, TIMEOUT or PERM) the result field may contain an explanatory message. However, the explanatory message is not used any further by *sendmail*.

Example replies:

    31:OK resolved.address@example.com,

    56:OK error:550 5.7.1 User does not accept mail from sender,

in case of successful lookups, or:

    8:NOTFOUND,

in case the key was not found, or:

    55:TEMP this text explains that we had a temporary failure,

in case of a temporary map lookup failure.

The socket map uses the same syntax as milters (see Section "X — Mail Filter (Milter) Definitions") to specify the remote endpoint, e.g.,

    Ksocket mySocketMap inet:12345@127.0.0.1

If multiple socket maps define the same remote endpoint, they will share a single connection to this endpoint.

Most of these 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, −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 **−N** is specified it never tries without a null byte and if **−O** is specified it never tries with a null byte. Setting one of these can speed matches but are 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.

−T*x*
: Append the string *x* on temporary failures. For example, *x* would be appended if a DNS lookup returned "server failed" or an NIS lookup could not locate a server. See also the **−t** flag.

−f
: Do not fold upper to lower case 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.

−k*keycol*
: The key column name (for NIS+) or number (for text lookups). For LDAP maps this is an LDAP filter string in which %s is replaced with the literal contents of the lookup key and %0 is replaced with the LDAP escaped contents of the lookup key according to RFC 2254. If the flag **−K** is used, then %1 through %9 are replaced with the LDAP escaped contents of the arguments specified in the map lookup.

−v*valcol*
: The value column name (for NIS+) or number (for text lookups). For LDAP maps this is the name of one or more attributes to be returned; multiple attributes can be separated by commas. If not specified, all attributes found in the match will be returned. The attributes listed can also include a type and one or more objectClass values for matching as described in the LDAP section.

−z*delim*
: The column delimiter (for text lookups). It can be a single character or one of the special strings "\n" or "\t" to indicate newline or tab respectively. If omitted entirely, the column separator is any sequence of white space. For LDAP maps this is the separator character to combine multiple values into a single return string. If not set, the LDAP lookup will only return the first match found. For DNS maps this is the separator character at which the result of a query is cut off if is too long.

−t
: Normally, when a map attempts to do a lookup and the server fails (e.g., *sendmail* couldn't contact any name server; this is *not* the same as an entry not being found in the map), the message being processed is queued for future processing. The **−t** flag turns off this behavior, letting the temporary failure (server down) act as though it were a permanent failure (entry not found). It is particularly useful for DNS lookups, where someone else's misconfigured name server can cause problems on your machine. However, care must be taken to ensure that you don't bounce mail that would be resolved correctly if you tried again. A common

strategy is to forward such mail to another, possibly better connected, mail server.

−D Perform no lookup in deferred delivery mode. This flag is set by default for the *host* map.

−S*spacesub* The character to use to replace space characters after a successful map lookup (esp. useful for regex and syslog maps).

−s*spacesub* For the dequote map only, the character to use to replace space characters after a successful dequote.

−q Don't dequote the key before lookup.

−L*level* For the syslog map only, it specifies the level to use for the syslog call.

−A When rebuilding an alias file, the **−A** flag causes duplicate entries in the text version to be merged. For example, two entries:

    list:    user1, user2
    list:    user3

would be treated as though it were the single entry

    list:    user1, user2, user3

in the presence of the **−A** flag.

Some additional flags are available for the host and dns maps:

−d delay: specify the resolver's retransmission time interval (in seconds).

−r retry: specify the number of times to retransmit a resolver query.

The dns map has another flag:

−B basedomain: specify a domain that is always appended to queries.

The following additional flags are present in the ldap map only:

−R Do not auto chase referrals. sendmail must be compiled with **−DLDAP_REFERRALS** to use this flag.

−n Retrieve attribute names only.

−V*sep* Retrieve both attributes name and value(s), separated by *sep*.

−r*deref* Set the alias dereference option to one of never, always, search, or find.

−s*scope* Set search scope to one of base, one (one level), or sub (subtree).

−h*host* LDAP server hostname. Some LDAP libraries allow you to specify multiple, space-separated hosts for redundancy. In addition, each of the hosts listed can be followed by a colon and a port number to override the default LDAP port.

−p*port* LDAP service port.

−H *LDAPURI* Use the specified LDAP URI instead of specifying the hostname and port separately with the the **−h** and **−p** options shown above. For example,

    -h server.example.com -p 389 -b dc=example,dc=com

is equivalent to

    -H ldap://server.example.com:389 -b dc=example,dc=com

If the LDAP library supports it, the LDAP URI format however can also request LDAP over SSL by using **ldaps://** instead of **ldap://**. For example:

    O LDAPDefaultSpec=-H ldaps://ldap.example.com -b dc=example,dc=com

Similarly, if the LDAP library supports it, It can also be used to specify a UNIX domain socket using **ldapi://**:

O LDAPDefaultSpec=-H ldapi://socketfile -b dc=example,dc=com

−b*base* LDAP search base.

−l*timelimit* Time limit for LDAP queries.

−Z*sizelimit* Size (number of matches) limit for LDAP or DNS queries.

−d*distinguished_name*
The distinguished name to use to login to the LDAP server.

−M*method* The method to authenticate to the LDAP server. Should be one of **LDAP_AUTH_NONE**, **LDAP_AUTH_SIMPLE**, or **LDAP_AUTH_KRBV4**.

−P*passwordfile* The file containing the secret key for the **LDAP_AUTH_SIMPLE** authentication method or the name of the Kerberos ticket file for **LDAP_AUTH_KRBV4**.

−1 Force LDAP searches to only succeed if a single match is found. If multiple values are found, the search is treated as if no match was found.

−w*version* Set the LDAP API/protocol version to use. The default depends on the LDAP client libraries in use. For example, **−w 3** will cause *sendmail* to use LDAPv3 when communicating with the LDAP server.

−K Treat the LDAP search key as multi-argument and replace %1 through %9 in the key with the LDAP escaped contents of the lookup arguments specified in the map lookup.

The *dbm* map appends the strings ".pag" and ".dir" to the given filename; the *hash* and *btree* maps append ".db". For example, the map specification

Kuucp dbm −o −N /etc/mail/uucpmap

specifies an optional map named "uucp" of class "dbm"; it always has null bytes at the end of every string, and the data is located in /etc/mail/uucpmap.{dir,pag}.

The program *makemap*(8) can be used to build any of the three database-oriented maps. It takes the following flags:

−f Do not fold upper to lower case 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 won't be read while they are being updated.

New classes can be added in the routine **setupmaps** in file **conf.c**.

### 5.10. Q — Queue Group Declaration

In addition to the option *QueueDirectory,* queue groups can be declared that define a (group of) queue directories under a common name. The syntax is as follows:

**Q***name* {, *field*=*value* }+

where *name* is the symbolic name of the queue group under which it can be referenced in various places and the "field=value" pairs define attributes of the queue group. The name must only consist of alphanumeric characters. Fields are:

Flags Flags for this queue group.

| | |
|---|---|
| Nice | The nice(2) increment for the queue group. This value must be greater or equal zero. |
| Interval | The time between two queue runs. |
| Path | The queue directory of the group (required). |
| Runners | The number of parallel runners processing the queue. Note that **F=f** must be set if this value is greater than one. |
| Jobs | The maximum number of jobs (messages delivered) per queue run. |
| recipients | The maximum number of recipients per envelope. Envelopes with more than this number of recipients will be split into multiple envelopes in the same queue directory. The default value 0 means no limit. |

Only the first character of the field name is checked.

By default, a queue group named *mqueue* is defined that uses the value of the *QueueDirectory* option as path. Notice: all paths that are used for queue groups must be subdirectories of *QueueDirectory*. Since they can be symbolic links, this isn't a real restriction, If *QueueDirectory* uses a wildcard, then the directory one level up is considered the ''base'' directory which all other queue directories must share. Please make sure that the queue directories do not overlap, e.g., do not specify

```
O QueueDirectory=/var/spool/mqueue/*
Qone, P=/var/spool/mqueue/dir1
Qtwo, P=/var/spool/mqueue/dir2
```

because this also includes "dir1" and "dir2" in the default queue group. However,

```
O QueueDirectory=/var/spool/mqueue/main*
Qone, P=/var/spool/mqueue/dir
Qtwo, P=/var/spool/mqueue/other*
```

is a valid queue group specification.

Options listed in the ''Flags'' field can be used to modify the behavior of a queue group. The ''f'' flag must be set if multiple queue runners are supposed to work on the entries in a queue group. Otherwise *sendmail* will work on the entries strictly sequentially.

The ''Interval'' field sets the time between queue runs. If no queue group specific interval is set, then the parameter of the **-q** option from the command line is used.

To control the overall number of concurrently active queue runners the option **MaxQueueChildren** can be set. This limits the number of processes used for running the queues to **MaxQueueChildren**, though at any one time fewer processes may be active as a result of queue options, completed queue runs, system load, etc.

The maximum number of queue runners for an individual queue group can be controlled via the **Runners** option. If set to 0, entries in the queue will not be processed, which is useful to ''quarantine'' queue files. The number of runners per queue group may also be set with the option **MaxRunnersPerQueue**, which applies to queue groups that have no individual limit. That is, the default value for **Runners** is **MaxRunnersPerQueue** if set, otherwise 1.

The field Jobs describes the maximum number of jobs (messages delivered) per queue run, which is the queue group specific value of **MaxQueueRunSize**.

Notice: queue groups should be declared after all queue related options have been set because queue groups take their defaults from those options. If an option is set after a queue group declaration, the values of options in the queue group are set to the defaults of *sendmail* unless explicitly set in the declaration.

Each envelope is assigned to a queue group based on the algorithm described in section ''Queue Groups and Queue Directories''.

### 5.11. X — Mail Filter (Milter) Definitions

The *sendmail* Mail Filter API (Milter) is designed to allow third-party programs access to mail messages as they are being processed in order to filter meta-information and content. They are declared in the configuration file as:

> **X***name* {, *field*=*value* }*

where *name* is the name of the filter (used internally only) and the "field=name" pairs define attributes of the filter. Also see the documentation for the **InputMailFilters** option for more information.

Fields are:

| | |
|---|---|
| Socket | The socket specification |
| Flags | Special flags for this filter |
| Timeouts | Timeouts for this filter |

Only the first character of the field name is checked (it's case-sensitive).

The socket specification is one of the following forms:

> **S=inet:** *port* @ *host*
>
> **S=inet6:** *port* @ *host*
>
> **S=local:** *path*

The first two describe an IPv4 or IPv6 socket listening on a certain *port* at a given *host* or IP address. The final form describes a named socket on the filesystem at the given *path*.

The following flags may be set in the filter description.

R Reject connection if filter unavailable.

T Temporary fail connection if filter unavailable.

If neither F=R nor F=T is specified, the message is passed through *sendmail* in case of filter errors as if the failing filters were not present.

The timeouts can be set using the four fields inside of the **T=** equate:

C Timeout for connecting to a filter. If set to 0, the system's *connect()* timeout will be used.

S Timeout for sending information from the MTA to a filter.

R Timeout for reading reply from the filter.

E Overall timeout between sending end-of-message to filter and waiting for the final acknowledgment.

Note the separator between each timeout field is a **';'**. The default values (if not set) are: **T=C:5m;S:10s;R:10s;E:5m** where **s** is seconds and **m** is minutes.

Examples:

```
Xfilter1, S=local:/var/run/f1.sock, F=R
Xfilter2, S=inet6:999@localhost, F=T, T=S:1s;R:1s;E:5m
Xfilter3, S=inet:3333@localhost, T=C:2m
```

### 5.12. The User Database

The user database is deprecated in favor of ''virtusertable'' and ''genericstable'' as explained in the file **cf/README**. If you have a version of *sendmail* with the user database package compiled in, the handling of sender and recipient addresses is modified.

The location of this database is controlled with the **UserDatabaseSpec** option.

### 5.12.1. 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. 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. See also *:default:mailname*. |
| mailsender | Changes any mail sent to this address to have the indicated envelope sender. This is intended for mailing lists, and will normally be the name of an appropriate -request address. It is very similar to the owner-*list* syntax in the alias file. |
| fullname | The full name of the user. |
| office-address | The office address for this user. |
| office-phone | The office phone number for this user. |
| office-fax | The office FAX number for this user. |
| home-address | The home address for this user. |
| home-phone | The home phone number for this user. |
| home-fax | The home FAX number for this user. |
| project | A (short) description of the project this person is affiliated with. In the University this is often just the name of their graduate advisor. |
| plan | A pointer to a file from which plan information can be gathered. |

As of this writing, only a few of these fields are actually being used by *sendmail*: *maildrop* and *mailname*. A *finger* program that uses the other fields is planned.

### 5.12.2. User database semantics

When the rewriting rules submit an address to the local mailer, the user name is passed through the alias file. If no alias is found (or if the alias points back to the same address), the name (with ":maildrop" appended) is then used as a key in the user database. If no match occurs (or if the maildrop points at the same address), forwarding is tried.

If the first token of the user name returned by ruleset 0 is an "@" sign, the user database lookup is skipped. The intent is that the user database will act as a set of defaults for a cluster (in our case, the Computer Science Division); mail sent to a specific machine should ignore these defaults.

When mail is sent, the name of the sending user is looked up in the database. If that user has a "mailname" record, the value of that record is used as their outgoing name. For example, I might have a record:

```
eric:mailname      Eric.Allman@CS.Berkeley.EDU
```

This would cause my outgoing mail to be sent as Eric.Allman.

If a "maildrop" is found for the user, 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, in our case we would set it to "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.

#### 5.12.3. Creating the database[23]

The user database is built from a text file using the *makemap* utility (in the distribution in the makemap subdirectory). The text file is a series of lines corresponding to userdb records; each line has a key and a value separated by white space. The key is always in the format described above — for example:

eric:maildrop

This file is normally installed in a system directory; for example, it might be called */etc/mail/userdb*. To make the database version of the map, run the program:

makemap btree /etc/mail/userdb < /etc/mail/userdb

Then create a config file that uses this. For example, using the V8 M4 configuration, include the following line in your .mc file:

define(`confUSERDB_SPEC´, /etc/mail/userdb)

## 6. OTHER CONFIGURATION

There are some configuration changes that can be made by recompiling *sendmail*. This section describes what changes can be made and what has to be modified to make them. In most cases this should be unnecessary unless you are porting *sendmail* to a new environment.

### 6.1. Parameters in devtools/OS/$oscf

These parameters are intended to describe the compilation environment, not site policy, and should normally be defined in the operating system configuration file. **This section needs a complete rewrite.**

NDBM — If set, the new version of the DBM library that allows multiple databases will be used. If neither NDBM nor NEWDB are set, a much less efficient method of alias lookup is used.

NEWDB — If set, use the new database package from Berkeley (from 4.4BSD). This package is substantially faster than DBM or NDBM. If NEWDB and NDBM are both set, *sendmail* will read DBM files, but will create and use NEWDB files.

NIS — Include support for NIS. If set together with *both* NEWDB and NDBM, *sendmail* will create both DBM and NEWDB files if and only if an alias file includes the substring "/yp/" in the name. This is intended for compatibility with Sun Microsystems' *mkalias* program used on YP masters.

NISPLUS — Compile in support for NIS+.

NETINFO — Compile in support for NetInfo (NeXT stations).

LDAPMAP — Compile in support for LDAP X500 queries. Requires libldap and liblber from the Umich LDAP 3.2 or 3.3 release or equivalent libraries for other LDAP libraries such as OpenLDAP.

HESIOD — Compile in support for Hesiod.

[23]These instructions are known to be incomplete. Other features are available which provide similar functionality, e.g., virtual hosting and mapping local addresses into a generic form as explained in cf/README.

MAP_NSD Compile in support for IRIX NSD lookups.

MAP_REGEX Compile in support for regular expression matching.

DNSMAP Compile in support for DNS map lookups in the *sendmail.cf* file.

PH_MAP Compile in support for ph lookups.

SASL Compile in support for SASL, a required component for SMTP Authentication support.

STARTTLS Compile in support for STARTTLS.

EGD Compile in support for the "Entropy Gathering Daemon" to provide better random data for TLS.

TCPWRAPPERS Compile in support for TCP Wrappers.

_PATH_SENDMAILCF
The pathname of the sendmail.cf file.

_PATH_SENDMAILPID
The pathname of the sendmail.pid file.

SM_CONF_SHM
Compile in support for shared memory, see section about "/var/spool/mqueue".

MILTER Compile in support for contacting external mail filters built with the Milter API.

There are also several compilation flags to indicate the environment such as "_AIX3" and "_SCO_unix_". See the sendmail/README file for the latest scoop on these flags.

### 6.2. Parameters in sendmail/conf.h

Parameters and compilation options are defined in conf.h. Most of these need not normally be tweaked; common parameters are all in sendmail.cf. However, the sizes of certain primitive vectors, etc., are included in this file. The numbers following the parameters are their default value.

This document is not the best source of information for compilation flags in conf.h — see sendmail/README or sendmail/conf.h itself.

MAXLINE [2048] The maximum line length of any input line. If message lines exceed this length they will still be processed correctly; however, header lines, configuration file lines, alias lines, etc., must fit within this limit.

MAXNAME [256] The maximum length of any name, such as a host or a user name.

MAXPV [256] The maximum number of parameters to any mailer. This limits the number of recipients that may be passed in one transaction. It can be set to any arbitrary number above about 10, since *sendmail* will break up a delivery into smaller batches as needed. A higher number may reduce load on your system, however.

MAXQUEUEGROUPS [50]
The maximum number of queue groups.

MAXATOM [1000] The maximum number of atoms (tokens) in a single address. For example, the address "eric@CS.Berkeley.EDU" is seven atoms.

MAXMAILERS [25] The maximum number of mailers that may be defined in the configuration file. This value is defined in include/sendmail/sendmail.h.

MAXRWSETS [200] The maximum number of rewriting sets that may be defined. The first half of these are reserved for numeric specification (e.g., "S92"), while the upper half are reserved for auto-numbering (e.g., "Sfoo"). Thus, with a value of 200 an attempt to use "S99" will succeed, but "S100" will fail.

MAXPRIORITIES [25]
: The maximum number of values for the “Precedence:” field that may be defined (using the **P** line in sendmail.cf).

MAXUSERENVIRON [100]
: The maximum number of items in the user environment that will be passed to subordinate mailers.

MAXMXHOSTS [100]
: The maximum number of MX records we will accept for any single host.

MAXMAPSTACK [12]
: The maximum number of maps that may be "stacked" in a **sequence** class map.

MAXMIMEARGS [20]
: The maximum number of arguments in a MIME Content-Type: header; additional arguments will be ignored.

MAXMIMENESTING [20]
: The maximum depth to which MIME messages may be nested (that is, nested Message or Multipart documents; this does not limit the number of components in a single Multipart document).

MAXDAEMONS [10]
: The maximum number of sockets sendmail will open for accepting connections on different ports.

MAXMACNAMELEN [25]
: The maximum length of a macro name.

A number of other compilation options exist. These specify whether or not specific code should be compiled in. Ones marked with † are 0/1 valued.

NETINET† — If set, support for Internet protocol networking is compiled in. Previous versions of *sendmail* referred to this as DAEMON; this old usage is now incorrect. Defaults on; turn it off in the Makefile if your system doesn’t support the Internet protocols.

NETINET6† — If set, support for IPv6 networking is compiled in. It must be separately enabled by adding **DaemonPortOptions** settings.

NETISO† — If set, support for ISO protocol networking is compiled in (it may be appropriate to #define this in the Makefile instead of conf.h).

NETUNIX† — If set, support for UNIX domain sockets is compiled in. This is used for control socket support.

LOG — If set, the *syslog* routine in use at some sites is used. This makes an informational log record for each message processed, and makes a higher priority log record for internal system errors. **STRONGLY RECOMMENDED** — if you want no logging, turn it off in the configuration file.

MATCHGECOS† — Compile in the code to do ‘‘fuzzy matching’’ on the GECOS field in /etc/passwd. This also requires that the **MatchGECOS** option be turned on.

NAMED_BIND† — Compile in code to use the Berkeley Internet Name Domain (BIND) server to resolve TCP/IP host names.

NOTUNIX — If you are using a non-UNIX mail format, you can set this flag to turn off special processing of UNIX-style “From ” lines.

USERDB† — Include the **experimental** Berkeley user information database package. This adds a new level of local name expansion between aliasing and forwarding. It also uses the NEWDB package. This may change in future releases.

The following options are normally turned on in per-operating-system clauses in conf.h.

IDENTPROTO† Compile in the IDENT protocol as defined in RFC 1413. This defaults on for all systems except Ultrix, which apparently has the interesting "feature" that when it receives a "host unreachable" message it closes all open connections to that host. Since some firewall gateways send this error code when you access an unauthorized port (such as 113, used by IDENT), Ultrix cannot receive email from such hosts.

SYSTEM5 Set all of the compilation parameters appropriate for System V.

HASFLOCK† Use Berkeley-style **flock** instead of System V **lockf** to do file locking. Due to the highly unusual semantics of locks across forks in **lockf**, this should always be used if at all possible.

HASINITGROUPS Set this if your system has the *initgroups()* call (if you have multiple group support). This is the default if SYSTEM5 is *not* defined or if you are on HPUX.

HASUNAME Set this if you have the *uname*(2) system call (or corresponding library routine). Set by default if SYSTEM5 is set.

HASGETDTABLESIZE
Set this if you have the *getdtablesize*(2) system call.

HASWAITPID Set this if you have the *haswaitpid*(2) system call.

FAST_PID_RECYCLE
Set this if your system can possibly reuse the same pid in the same second of time.

SFS_TYPE The mechanism that can be used to get file system capacity information. The values can be one of SFS_USTAT (use the ustat(2) syscall), SFS_4ARGS (use the four argument statfs(2) syscall), SFS_VFS (use the two argument statfs(2) syscall including <sys/vfs.h>), SFS_MOUNT (use the two argument statfs(2) syscall including <sys/mount.h>), SFS_STATFS (use the two argument statfs(2) syscall including <sys/statfs.h>), SFS_STATVFS (use the two argument statfs(2) syscall including <sys/statvfs.h>), or SFS_NONE (no way to get this information).

LA_TYPE The load average type. Details are described below.

The are several built-in ways of computing the load average. *Sendmail* tries to auto-configure them based on imperfect guesses; you can select one using the *cc* option **–DLA_TYPE=***type*, where *type* is:

LA_INT The kernel stores the load average in the kernel as an array of long integers. The actual values are scaled by a factor FSCALE (default 256).

LA_SHORT The kernel stores the load average in the kernel as an array of short integers. The actual values are scaled by a factor FSCALE (default 256).

LA_FLOAT The kernel stores the load average in the kernel as an array of double precision floats.

LA_MACH Use MACH-style load averages.

LA_SUBR Call the *getloadavg* routine to get the load average as an array of doubles.

LA_ZERO Always return zero as the load average. This is the fallback case.

If type LA_INT, LA_SHORT, or LA_FLOAT is specified, you may also need to specify _PATH_UNIX (the path to your system binary) and LA_AVENRUN (the name of the variable containing the load average in the kernel; usually "_avenrun" or "avenrun").

## 6.3. Configuration in sendmail/conf.c

The following changes can be made in conf.c.

### 6.3.1. Built-in Header Semantics

Not all header semantics are defined in the configuration file. Header lines that should only be included by certain mailers (as well as other more obscure semantics) must be specified in the *HdrInfo* table in *conf.c*. This table contains the header name (which should be in all lower case) and a set of header control flags (described below), The flags are:

H_ACHECK — Normally when the check is made to see if a header line is compatible with a mailer, *sendmail* will not delete an existing line. If this flag is set, *sendmail* will delete even existing header lines. That is, if this bit is set and the mailer does not have flag bits set that intersect with the required mailer flags in the header definition in sendmail.cf, the header line is *always* deleted.

H_EOH — If this header field is set, treat it like a blank line, i.e., it will signal the end of the header and the beginning of the message text.

H_FORCE — Add this header entry even if one existed in the message before. If a header entry does not have this bit set, *sendmail* will not add another header line if a header line of this name already existed. This would normally be used to stamp the message by everyone who handled it.

H_TRACE — If set, this is a timestamp (trace) field. If the number of trace fields in a message exceeds a preset amount the message is returned on the assumption that it has an aliasing loop.

H_RCPT — If set, this field contains recipient addresses. This is used by the **−t** flag to determine who to send to when it is collecting recipients from the message.

H_FROM — This flag indicates that this field specifies a sender. The order of these fields in the *HdrInfo* table specifies *sendmail*'s preference for which field to return error messages to.

H_ERRORSTO — Addresses in this header should receive error messages.

H_CTE — This header is a Content-Transfer-Encoding header.

H_CTYPE — This header is a Content-Type header.

H_STRIPVAL — Strip the value from the header (for Bcc:).

Let's look at a sample *HdrInfo* specification:

```
struct hdrinfo                  HdrInfo[] =
{
        /* originator fields, most to least significant  */
    "resent-sender",            H_FROM,
    "resent-from",              H_FROM,
    "sender",                   H_FROM,
    "from",                     H_FROM,
    "full-name",                H_ACHECK,
    "errors-to",                H_FROM|H_ERRORSTO,
        /* destination fields */
    "to",                       H_RCPT,
    "resent-to",                H_RCPT,
    "cc",                       H_RCPT,
    "bcc",                      H_RCPT|H_STRIPVAL,
        /* message identification and control */
    "message",                  H_EOH,
    "text",                     H_EOH,
        /* trace fields */
    "received",                 H_TRACE|H_FORCE,
        /* miscellaneous fields */
    "content-transfer-encoding", H_CTE,
    "content-type",             H_CTYPE,

    NULL,                       0,
};
```

This structure indicates that the “To:”, “Resent-To:”, and “Cc:” fields all specify recipient addresses. Any “Full-Name:” field will be deleted unless the required mailer flag (indicated in the configuration file) is specified. The “Message:” and “Text:” fields will terminate the header; these are used by random dissenters around the network world. The “Received:” field will always be added, and can be used to trace messages.

There are a number of important points here. First, header fields are not added automatically just because they are in the *HdrInfo* structure; they must be specified in the configuration file in order to be added to the message. Any header fields mentioned in the configuration file but not mentioned in the *HdrInfo* structure have default processing performed; that is, they are added unless they were in the message already. Second, the *HdrInfo* structure only specifies cliched processing; certain headers are processed specially by ad hoc code regardless of the status specified in *HdrInfo*. For example, the “Sender:” and “From:” fields are always scanned on ARPANET mail to determine the sender[24]; this is used to perform the “return to sender” function. The “From:” and “Full-Name:” fields are used to determine the full name of the sender if possible; this is stored in the macro **$x** and used in a number of ways.

### 6.3.2. Restricting Use of Email

If it is necessary to restrict mail through a relay, the *checkcompat* routine can be modified. This routine is called for every recipient address. It returns an exit status indicating the status of the message. The status EX_OK accepts the address, EX_TEMPFAIL queues the message for a later try, and other values (commonly EX_UNAVAILABLE) reject the message. It is up to *checkcompat* to print an error message (using *usrerr*) if the message is rejected. For example,

[24]Actually, this is no longer true in SMTP; this information is contained in the envelope. The older ARPANET protocols did not completely distinguish envelope from header.

*checkcompat* could read:

```
int
checkcompat(to, e)
	register ADDRESS *to;
	register ENVELOPE *e;
{
	register STAB *s;

	s = stab("private", ST_MAILER, ST_FIND);
	if (s != NULL && e->e_from.q_mailer != LocalMailer &&
	    to->q_mailer == s->s_mailer)
	{
		usrerr("No private net mail allowed through this machine");
		return (EX_UNAVAILABLE);
	}
	if (MsgSize > 50000 && bitnset(M_LOCALMAILER, to->q_mailer))
	{
		usrerr("Message too large for non-local delivery");
		e->e_flags |= EF_NORETURN;
		return (EX_UNAVAILABLE);
	}
	return (EX_OK);
}
```

This would reject messages greater than 50000 bytes unless they were local. The *EF_NORETURN* flag can be set in *e→e_flags* to suppress the return of the actual body of the message in the error return. The actual use of this routine is highly dependent on the implementation, and use should be limited.

#### 6.3.3. New Database Map Classes

New key maps can be added by creating a class initialization function and a lookup function. These are then added to the routine *setupmaps.*

The initialization function is called as

*xxx*_map_init(MAP *map, char *args)

The *map* is an internal data structure. The *args* is a pointer to the portion of the configuration file line following the map class name; flags and filenames can be extracted from this line. The initialization function must return true if it successfully opened the map, false otherwise.

The lookup function is called as

*xxx*_map_lookup(MAP *map, char buf[], char **av, int *statp)

The *map* defines the map internally. The *buf* has the input key. This may be (and often is) used destructively. The *av* is a list of arguments passed in from the rewrite line. The lookup function should return a pointer to the new value. If the map lookup fails, **statp* should be set to an exit status code; in particular, it should be set to EX_TEMPFAIL if recovery is to be attempted by the higher level code.

#### 6.3.4. Queueing Function

The routine *shouldqueue* is called to decide if a message should be queued or processed immediately. Typically this compares the message priority to the current load average. The default definition is:

```
bool
shouldqueue(pri, ctime)
	long pri;
	time_t ctime;
{
	if (CurrentLA < QueueLA)
		return false;
	return (pri > (QueueFactor / (CurrentLA − QueueLA + 1)));
}
```

If the current load average (global variable *CurrentLA*, which is set before this function is called) is less than the low threshold load average (option **x**, variable *QueueLA*), *shouldqueue* returns false immediately (that is, it should *not* queue). If the current load average exceeds the high threshold load average (option **X**, variable *RefuseLA*), *shouldqueue* returns true immediately. Otherwise, it computes the function based on the message priority, the queue factor (option **q**, global variable *QueueFactor*), and the current and threshold load averages.

An implementation wishing to take the actual age of the message into account can also use the *ctime* parameter, which is the time that the message was first submitted to *sendmail*. Note that the *pri* parameter is already weighted by the number of times the message has been tried (although this tends to lower the priority of the message with time); the expectation is that the *ctime* would be used as an "escape clause" to ensure that messages are eventually processed.

#### 6.3.5. Refusing Incoming SMTP Connections

The function *refuseconnections* returns true if incoming SMTP connections should be refused. The current implementation is based exclusively on the current load average and the refuse load average option (option **X**, global variable *RefuseLA*):

```
bool
refuseconnections()
{
	return (RefuseLA > 0 && CurrentLA >= RefuseLA);
}
```

A more clever implementation could look at more system resources.

#### 6.3.6. Load Average Computation

The routine *getla* returns the current load average (as a rounded integer). The distribution includes several possible implementations. If you are porting to a new environment you may need to add some new tweaks.[25]

### 6.4. Configuration in sendmail/daemon.c

The file *sendmail/daemon.c* contains a number of routines that are dependent on the local networking environment. The version supplied assumes you have BSD style sockets.

In previous releases, we recommended that you modify the routine *maphostname* if you wanted to generalize **$[** ... **$]** lookups. We now recommend that you create a new keyed map instead.

---

[25] If you do, please send updates to sendmail@Sendmail.ORG.

## 6.5. LDAP

In this section we assume that *sendmail* has been compiled with support for LDAP.

### 6.5.1. LDAP Recursion

LDAP Recursion allows you to add types to the search attributes on an LDAP map specification. The syntax is:

−v *ATTRIBUTE*[:*TYPE*[:*OBJECTCLASS*[|*OBJECTCLASS*|...]]]

The new *TYPE*s are:

NORMAL — This attribute type specifies the attribute to add to the results string. This is the default.

DN — Any matches for this attribute are expected to have a value of a fully qualified distinguished name. *sendmail* will lookup that DN and apply the attributes requested to the returned DN record.

FILTER — Any matches for this attribute are expected to have a value of an LDAP search filter. *sendmail* will perform a lookup with the same parameters as the original search but replaces the search filter with the one specified here.

URL — Any matches for this attribute are expected to have a value of an LDAP URL. *sendmail* will perform a lookup of that URL and use the results from the attributes named in that URL. Note however that the search is done using the current LDAP connection, regardless of what is specified as the scheme, LDAP host, and LDAP port in the LDAP URL.

Any untyped attributes are considered NORMAL attributes as described above.

The optional *OBJECTCLASS* (| separated) list contains the objectClass values for which that attribute applies. If the list is given, the attribute named will only be used if the LDAP record being returned is a member of that object class. Note that if these new value attribute *TYPE*s are used in an AliasFile option setting, it will need to be double quoted to prevent *sendmail* from misparsing the colons.

Note that LDAP recursion attributes which do not ultimately point to an LDAP record are not considered an error.

#### 6.5.1.1. Example

Since examples usually help clarify, here is an example which uses all four of the new types:

```
O LDAPDefaultSpec=-h ldap.example.com -b dc=example,dc=com

Kexample ldap
        -z,
        -k (&(objectClass=sendmailMTAAliasObject)(sendmailMTAKey=%0))
        -v sendmailMTAAliasValue,mail:NORMAL:inetOrgPerson,
           uniqueMember:DN:groupOfUniqueNames,
           sendmailMTAAliasSearch:FILTER:sendmailMTAAliasObject,
           sendmailMTAAliasURL:URL:sendmailMTAAliasObject
```

That definition specifies that:

- Any value in a sendmailMTAAliasValue attribute will be added to the result string regardless of object class.
- The mail attribute will be added to the result string if the LDAP record is a member of the inetOrgPerson object class.

- The uniqueMember attribute is a recursive attribute, used only in groupOfUniqueNames records, and should contain an LDAP DN pointing to another LDAP record. The desire here is to return the mail attribute from those DNs.
- The sendmailMTAAliasSearch attribute and sendmailMTAAliasURL are both used only if referenced in a sendmailMTAAliasObject. They are both recursive, the first for a new LDAP search string and the latter for an LDAP URL.

## 6.6. STARTTLS

In this section we assume that *sendmail* has been compiled with support for STARTTLS. To properly understand the use of STARTTLS in *sendmail*, it is necessary to understand at least some basics about X.509 certificates and public key cryptography. This information can be found in books about SSL/TLS or on WWW sites, e.g., “http://www.OpenSSL.org/”.

### 6.6.1. Certificates for STARTTLS

When acting as a server, *sendmail* requires X.509 certificates to support STARTTLS: one as certificate for the server (ServerCertFile and corresponding private ServerKeyFile) at least one root CA (CACertFile), i.e., a certificate that is used to sign other certificates, and a path to a directory which contains other CAs (CACertPath). The file specified via CACertFile can contain several certificates of CAs. The DNs of these certificates are sent to the client during the TLS handshake (as part of the CertificateRequest) as the list of acceptable CAs. However, do not list too many root CAs in that file, otherwise the TLS handshake may fail; e.g.,

```
error:14094417:SSL routines:SSL3_READ_BYTES:
sslv3 alert illegal parameter:s3_pkt.c:964:SSL alert number 47
```

You should probably put only the CA cert into that file that signed your own cert(s), or at least only those you trust. The CACertPath directory must contain the hashes of each CA certificate as filenames (or as links to them). Symbolic links can be generated with the following two (Bourne) shell commands:

```
C=FileName_of_CA_Certificate
ln -s $C `openssl x509 -noout -hash < $C`.0
```

An X.509 certificate is also required for authentication in client mode (ClientCertFile and corresponding private ClientKeyFile), however, *sendmail* will always use STARTTLS when offered by a server. The client and server certificates can be identical. Certificates can be obtained from a certificate authority or created with the help of OpenSSL. The required format for certificates and private keys is PEM. To allow for automatic startup of sendmail, private keys (ServerKeyFile, ClientKeyFile) must be stored unencrypted. The keys are only protected by the permissions of the file system. Never make a private key available to a third party.

### 6.6.2. PRNG for STARTTLS

STARTTLS requires a strong pseudo random number generator (PRNG) to operate properly. Depending on the TLS library you use, it may be required to explicitly initialize the PRNG with random data. OpenSSL makes use of **/dev/urandom(4)** if available (this corresponds to the compile flag HASURANDOMDEV). On systems which lack this support, a random file must be specified in the *sendmail.cf* file using the option RandFile. It is **strongly** advised to use the "Entropy Gathering Daemon" EGD from Brian Warner on those systems to provide useful random data. In this case, *sendmail* must be compiled with the flag EGD, and the RandFile option must point to the EGD socket. If neither **/dev/urandom(4)** nor EGD are available, you have to make sure that useful random data is available all the time in RandFile. If the file hasn't been modified in the last 10 minutes before it is supposed to be used by *sendmail* the content is considered obsolete. One method for generating this file is:

openssl rand -out /etc/mail/randfile -rand */path/to/file:...*256

See the OpenSSL documentation for more information. In this case, the PRNG for TLS is only seeded with other random data if the **DontBlameSendmail** option **InsufficientEntropy** is set. This is most likely not sufficient for certain actions, e.g., generation of (temporary) keys.

Please see the OpenSSL documentation or other sources for further information about certificates, their creation and their usage, the importance of a good PRNG, and other aspects of TLS.

### 6.7. Encoding of STARTTLS and AUTH related Macros

Macros that contain STARTTLS and AUTH related data which comes from outside sources, e.g., all macros containing information from certificates, are encoded to avoid problems with non-printable or special characters. The latter are '\', '<', '>', '(', ')', '"', '+', and ' '. All of these characters are replaced by their value in hexadecimal with a leading '+'. For example:

/C=US/ST=California/O=endmail.org/OU=private/CN=Darth Mail (Cert)/
Email=darth+cert@endmail.org

is encoded as:

/C=US/ST=California/O=endmail.org/OU=private/
CN=Darth+20Mail+20+28Cert+29/Email=darth+2Bcert@endmail.org

(line breaks have been inserted for readability). The macros which are subject to this encoding are {cert_subject}, {cert_issuer}, {cn_subject}, {cn_issuer}, as well as {auth_authen} and {auth_author}.

## 7. ACKNOWLEDGEMENTS

I've worked on *sendmail* for many years, and many employers have been remarkably patient about letting me work on a large project that was not part of my official job. This includes time on the INGRES Project at the University of California at Berkeley, at Britton Lee, and again on the Mammoth and Titan Projects at Berkeley.

Much of the second wave of improvements resulting in version 8.1 should be credited to Bryan Costales of the International Computer Science Institute. As he passed me drafts of his book on *sendmail* I was inspired to start working on things again. Bryan was also available to bounce ideas off of.

Gregory Neil Shapiro of Worcester Polytechnic Institute has become instrumental in all phases of *sendmail* support and development, and was largely responsible for getting versions 8.8 and 8.9 out the door.

Many, many people contributed chunks of code and ideas to *sendmail*. It has proven to be a group network effort. Version 8 in particular was a group project. The following people and organizations made notable contributions:

Claus Assmann
John Beck, Hewlett-Packard & Sun Microsystems
Keith Bostic, CSRG, University of California, Berkeley
Andrew Cheng, Sun Microsystems
Michael J. Corrigan, University of California, San Diego
Bryan Costales, International Computer Science Institute & InfoBeat
Pär (Pell) Emanuelsson
Craig Everhart, Transarc Corporation
Per Hedeland, Ericsson
Tom Ivar Helbekkmo, Norwegian School of Economics
Kari Hurtta, Finnish Meteorological Institute
Allan E. Johannesen, WPI
Jonathan Kamens, OpenVision Technologies, Inc.

Takahiro Kanbe, Fuji Xerox Information Systems Co., Ltd.
Brian Kantor, University of California, San Diego
John Kennedy, Cal State University, Chico
Murray S. Kucherawy, HookUp Communication Corp.
Bruce Lilly, Sony U.S.
Karl London
Motonori Nakamura, Ritsumeikan University & Kyoto University
John Gardiner Myers, Carnegie Mellon University
Neil Rickert, Northern Illinois University
Gregory Neil Shapiro, WPI
Eric Schnoebelen, Convex Computer Corp.
Eric Wassenaar, National Institute for Nuclear and High Energy Physics, Amsterdam
Randall Winchester, University of Maryland
Christophe Wolfhugel, Pasteur Institute & Herve Schauer Consultants (Paris)
Exactis.com, Inc.

I apologize for anyone I have omitted, misspelled, misattributed, or otherwise missed. At this point, I suspect that at least a hundred people have contributed code, and many more have contributed ideas, comments, and encouragement. I've tried to list them in the RELEASE_NOTES in the distribution directory. I appreciate their contribution as well.

Special thanks are reserved for Michael Corrigan and Christophe Wolfhugel, who besides being wonderful guinea pigs and contributors have also consented to be added to the ''sendmail@Sendmail.ORG'' list and, by answering the bulk of the questions sent to that list, have freed me up to do other work.

# APPENDIX A

## COMMAND LINE FLAGS

Arguments must be presented with flags before addresses. The flags are:

| | |
|---|---|
| −A*x* | Select an alternative .cf file which is either *sendmail.cf* for **−Am** or *submit.cf* for **−Ac**. By default the .cf file is chosen based on the operation mode. For **-bm** (default), **-bs**, and **-t** it is *submit.cf* if it exists, for all others it is *sendmail.cf*. |
| −b*x* | Set operation mode to *x*. Operation modes are: |

| | |
|---|---|
| m | Deliver mail (default) |
| s | Speak SMTP on input side |
| a† | ‘‘Arpanet’’ mode (get envelope sender information from header) |
| d | Run as a daemon in background |
| D | Run as a daemon in foreground |
| t | Run in test mode |
| v | Just verify addresses, don’t collect or deliver |
| i | Initialize the alias database |
| p | Print the mail queue |
| P | Print overview over the mail queue (requires shared memory) |
| h | Print the persistent host status database |
| H | Purge expired entries from the persistent host status database |

| | |
|---|---|
| −B*type* | Indicate body type. |
| −C*file* | Use a different configuration file. *Sendmail* runs as the invoking user (rather than root) when this flag is specified. |
| −D *logfile* | Send debugging output to the indicated *logfile* instead of stdout. |
| −d*level* | Set debugging level. |
| −f *addr* | The envelope sender address is set to *addr*. This address may also be used in the From: header if that header is missing during initial submission. The envelope sender address is used as the recipient for delivery status notifications and may also appear in a Return-Path: header. |
| −F *name* | Sets the full name of this user to *name*. |
| −G | When accepting messages via the command line, indicate that they are for relay (gateway) submission. sendmail may complain about syntactically invalid messages, e.g., unqualified host names, rather than fixing them when this flag is set. sendmail will not do any canonicalization in this mode. |
| −h *cnt* | Sets the “hop count” to *cnt*. This represents the number of times this message has been processed by *sendmail* (to the extent that it is supported by the underlying networks). *Cnt* is incremented during processing, and if it reaches MAXHOP (currently 25) *sendmail* throws away the message with an error. |

†Deprecated.

−L *tag* Sets the identifier used for syslog. Note that this identifier is set as early as possible. However, *sendmail* may be used if problems arise before the command line arguments are processed.

−n Don't do aliasing or forwarding.

−N *notifications* Tag all addresses being sent as wanting the indicated *notifications*, which consists of the word "NEVER" or a comma-separated list of "SUCCESS", "FAILURE", and "DELAY" for successful delivery, failure, and a message that is stuck in a queue somewhere. The default is "FAILURE,DELAY".

−r *addr* An obsolete form of **−f**.

−o*x value* Set option *x* to the specified *value*. These options are described in Section 5.6.

−O*option***=***value* Set *option* to the specified *value* (for long form option names). These options are described in Section 5.6.

−M*x value* Set macro *x* to the specified *value*.

−p*protocol* Set the sending protocol. Programs are encouraged to set this. The protocol field can be in the form *protocol***:***host* to set both the sending protocol and sending host. For example, "−pUUCP:uunet" sets the sending protocol to UUCP and the sending host to uunet. (Some existing programs use −oM to set the r and s macros; this is equivalent to using −p.)

−q*time* Try to process the queued up mail. If the time is given, a *sendmail* will start one or more processes to run through the queue(s) at the specified time interval to deliver queued mail; otherwise, it only runs once. Each of these processes acts on a workgroup. These processes are also known as workgroup processes or WGP's for short. Each workgroup is responsible for controlling the processing of one or more queues; workgroups help manage the use of system resources by sendmail. Each workgroup may have one or more children concurrently processing queues depending on the setting of *MaxQueueChildren*.

−qp*time* Similar to −q with a time argument, except that instead of periodically starting WGP's sendmail starts persistent WGP's that alternate between processing queues and sleeping. The sleep time is specified by the time argument; it defaults to 1 second, except that a WGP always sleeps at least 5 seconds if their queues were empty in the previous run. Persistent processes are managed by a queue control process (QCP). The QCP is the parent process of the WGP's. Typically the QCP will be the sendmail daemon (when started with −bd or −bD) or a special process (named Queue control) (when started without −bd or −bD). If a persistent WGP ceases to be active for some reason another WGP will be started by the QCP for the same workgroup in most cases. When a persistent WGP has core dumped, the debug flag *no_persistent_restart* is set or the specific persistent WGP has been restarted too many times already then the WGP will not be started again and a message will be logged to this effect. To stop (SIGTERM) or restart (SIGHUP) persistent WGP's the appropriate signal should be sent to the QCP. The QCP will propagate the signal to all of the WGP's and if appropriate restart the persistent WGP's.

−q*Gname* Run the jobs in the queue group *name* once.

−q[!]*Xstring* Run the queue once, limiting the jobs to those matching *Xstring*. The key letter *X* can be **I** to limit based on queue identifier, **R** to limit based on recipient, **S** to limit based on sender, or **Q** to limit based on quarantine reason for quarantined jobs. A particular queued job is accepted if one of the corresponding attributes contains the indicated *string*. The optional ! character negates the condition tested. Multiple *−qX* flags are permitted, with items with the same key letter "or'ed" together, and items with different key letters "and'ed" together.

−Q[reason] Quarantine a normal queue items with the given reason or unquarantine quarantined queue items if no reason is given. This should only be used with some sort of item matching using **−q[!]***Xstring* as described above.

−R ret
: What information you want returned if the message bounces; *ret* can be “HDRS” for headers only or “FULL” for headers plus body. This is a request only; the other end is not required to honor the parameter. If “HDRS” is specified local bounces also return only the headers.

−t
: Read the header for “To:”, “Cc:”, and “Bcc:” lines, and send to everyone listed in those lists. The “Bcc:” line will be deleted before sending. Any addresses in the argument vector will be deleted from the send list.

−V envid
: The indicated *envid* is passed with the envelope of the message and returned if the message bounces.

−X *logfile*
: Log all traffic in and out of *sendmail* in the indicated *logfile* for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly.

There are a number of options that may be specified as primitive flags. These are the e, i, m, and v options. Also, the f option may be specified as the **−s** flag. The DSN related options “−N”, “−R”, and “−V” have no effects on *sendmail* running as daemon.

# APPENDIX B

# QUEUE FILE FORMATS

This appendix describes the format of the queue files. These files live in a queue directory. The individual qf, hf, Qf, df, and xf files may be stored in separate *qf/*, *df/*, and *xf/* subdirectories if they are present in the queue directory.

All queue files have the name *ttYMDhmsNNppppp* where *YMDhmsNNppppp* is the *id* for this message and the *tt* is a type. The individual letters in the *id* are:

Y	Encoded year

M	Encoded month

D	Encoded day

h	Encoded hour

m	Encoded minute

s	Encoded second

NN	Encoded envelope number

ppppp	At least five decimal digits of the process ID

All files with the same id collectively define one message. Due to the use of memory-buffered files, some of these files may never appear on disk.

The types are:

qf	The queue control file. This file contains the information necessary to process the job.

hf	The same as a queue control file, but for a quarantined queue job.

df	The data file. The message body (excluding the header) is kept in this file. Sometimes the df file is not stored in the same directory as the qf file; in this case, the qf file contains a 'd' record which names the queue directory that contains the df file.

tf	A 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.

xf	A transcript file, existing during the life of a session showing everything that happens during that session. Sometimes the xf file must be generated before a queue group has been selected; in this case, the xf file will be stored in a directory of the default queue group.

Qf	A ''lost'' queue control file. *sendmail* renames a **qf** file to **Qf** if there is a severe (configuration) problem that cannot be solved without human intervention. Search the logfile for the queue file id to figure out what happened. After you resolved the problem, you can rename the **Qf** file to **qf** and send it again.

The queue control file is structured as a series of lines each beginning with a code letter. The lines are as follows:

V	The version number of the queue file format, used to allow new *sendmail* binaries to read queue files created by older versions. Defaults to version zero. Must be the first line of the file if present. For 8.12 the version number is 6.

A	The information given by the AUTH= parameter of the "MAIL FROM:" command or $f@$j if sendmail has been called directly.

H A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

C The controlling address. The syntax is "localuser:aliasname". Recipient addresses following this line will be flagged so that deliveries will be run as the *localuser* (a user name from the /etc/passwd file); *aliasname* is the name of the alias that expanded to this address (used for printing messages).

q The quarantine reason for quarantined queue items.

Q The ''original recipient'', specified by the ORCPT= field in an ESMTP transaction. Used exclusively for Delivery Status Notifications. It applies only to the following 'R' line.

r The ''final recipient'' used for Delivery Status Notifications. It applies only to the following 'R' line.

R A recipient address. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient. Version 1 qf files also include a leading colon-terminated list of flags, which can be 'S' to return a message on successful final delivery, 'F' to return a message on failure, 'D' to return a message if the message is delayed, 'B' to indicate that the body should be returned, 'N' to suppress returning the body, and 'P' to declare this as a ''primary'' (command line or SMTP-session) address.

S The sender address. There may only be one of these lines.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order 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 A 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. Other flag bits are: **8**: the body contains 8bit data, **b**: a Bcc: header should be removed, **d**: the mail has RET parameters (see RFC 1894), **n**: the body of the message should not be returned in case of an error, **s**: the envelope has been split.

N The total number of delivery attempts.

K The time (as seconds since January 1, 1970) of the last delivery attempt.

d If the df file is in a different directory than the qf file, then a 'd' record is present, specifying the directory in which the df file resides.

I The i-number of the data file; this can be used to recover your mail queue after a disastrous disk crash.

$ A macro definition. The values of certain macros 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".

Z The original envelope id (from the ESMTP transaction). For Deliver Status Notifications only.

As an example, the following is a queue file sent to "eric@mammoth.Berkeley.EDU" and “bostic@okeeffe.CS.Berkeley.EDU”[1]:

```
V4
T711358135
K904446490
N0
P2100941
$_eric@localhost
${daemon_flags}
Seric
Ceric:100:1000:sendmail@vangogh.CS.Berkeley.EDU
RPFD:eric@mammoth.Berkeley.EDU
RPFD:bostic@okeeffe.CS.Berkeley.EDU
H?P?Return-path: <^g>
H??Received: by vangogh.CS.Berkeley.EDU (5.108/2.7) id AAA06703;
      Fri, 17 Jul 1992 00:28:55 -0700
H??Received: from mail.CS.Berkeley.EDU by vangogh.CS.Berkeley.EDU (5.108/2.7)
      id AAA06698; Fri, 17 Jul 1992 00:28:54 -0700
H??Received: from [128.32.31.21] by mail.CS.Berkeley.EDU (5.96/2.5)
      id AA22777; Fri, 17 Jul 1992 03:29:14 -0400
H??Received: by foo.bar.baz.de (5.57/Ultrix3.0-C)
      id AA22757; Fri, 17 Jul 1992 09:31:25 GMT
H?F?From: eric@foo.bar.baz.de (Eric Allman)
H?x?Full-name: Eric Allman
H??Message-id: <9207170931.AA22757@foo.bar.baz.de>
H??To: sendmail@vangogh.CS.Berkeley.EDU
H??Subject: this is an example message
```

This shows 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.

[1]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.

# APPENDIX C

## 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; check there to find the actual pathnames.

/usr/sbin/sendmail
: The binary of *sendmail*.

/usr/bin/newaliases
: A link to /usr/sbin/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*.

/etc/mail/sendmail.cf
: The configuration file, in textual form.

/etc/mail/helpfile
: The SMTP help file.

/etc/mail/statistics
: A statistics file; need not be present.

/etc/mail/sendmail.pid
: Created in daemon mode; it contains the process id of the current SMTP daemon. If you use this in scripts; use ‘‘head −1’’ to get just the first line; the second line contains the command line used to invoke the daemon, and later versions of *sendmail* may add more information to subsequent lines.

/etc/mail/aliases
: The textual version of the alias file.

/etc/mail/aliases.db
: The alias file in *hash* (3) format.

/etc/mail/aliases.{pag,dir}
: The alias file in *ndbm* (3) format.

/var/spool/mqueue
: The directory in which the mail queue(s) and temporary files reside.

/var/spool/mqueue/qf*
: Control (queue) files for messages.

/var/spool/mqueue/df*
: Data files.

/var/spool/mqueue/tf*
: Temporary versions of the qf files, used during queue file rebuild.

/var/spool/mqueue/xf*
: A transcript of the current session.

This page intentionally left blank;
replace it with a blank sheet for double-sided output.

# TABLE OF CONTENTS