APPENDIX A

How qmail Works

You don't need to understand how qmail works to install or use qmail. And you don't have to be an auto mechanic to operate a car or a watchmaker to tell time. But if you really want to master qmail, knowing exactly how it does what it does is crucial.

Luckily, qmail's simple, modular design makes understanding how it works easy for a system as complex as a Mail Transfer Agent (MTA). This appendix takes a top-down approach: first looking at how the modules interact with each other, then looking at how each module does its job.

High-Level Overview

The grand division in qmail is between the modules that accept new messages and place them into the queue and the modules that deliver them from the queue. We'll call these functions *receiving* and *sending*. The separation between receiving and sending is complete: Either of these functions can be fully operational while the other is shut down. Figure A-1 shows the high-level organization of qmail.

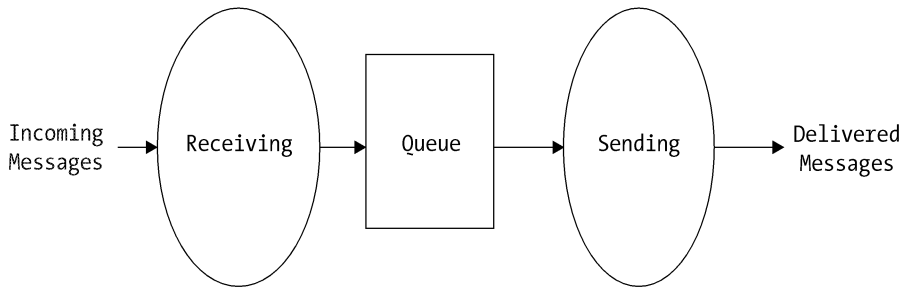

Figure A-1. High-level qmail organization

Receiving

Messages enter the queue through two main routes: local injection using `qmail-inject` or `sendmail` and network injection using `qmail-smtpd`, `qmail-qmtpd`

or qmail-qmqpd. Both of these routes use qmail-queue to actually inject their messages into the queue. Figure A-2 shows the organization of the receiving function.

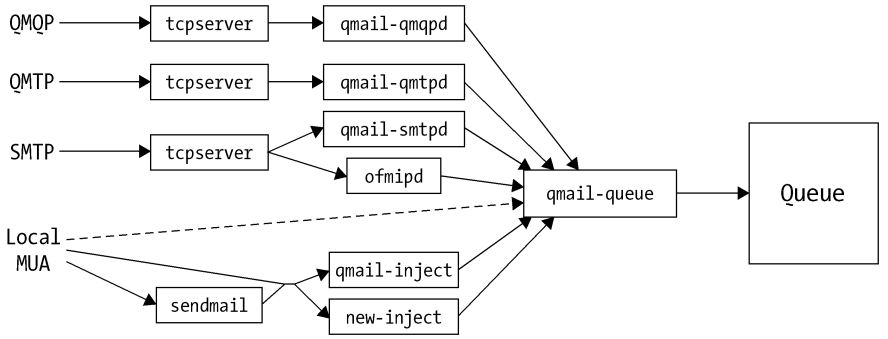

Figure A-2. The receiving function

Sending

Messages are delivered from the queue through two main routes: local delivery using qmail-local and remote delivery using qmail-remote. Both types of deliveries are dispatched by qmail-send through qmail-lspawn and qmail-rspawn, respectively. Figure A-3 shows the organization of the sending function.

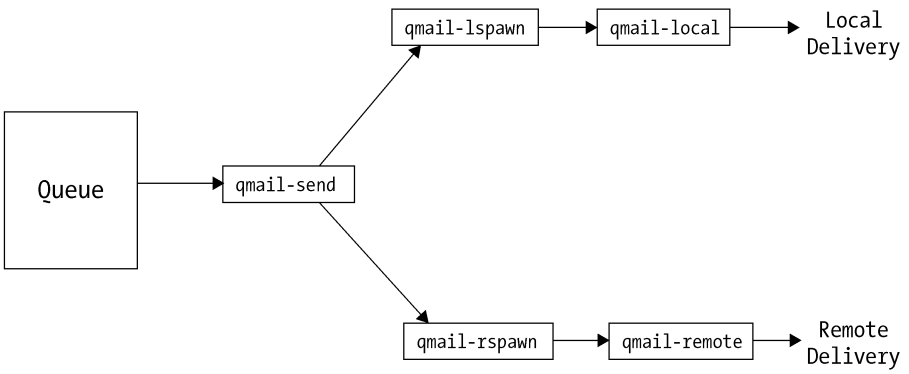

Figure A-3. The sending function

Receiving Modules

First we'll look at the modules comprising the receiving function: sendmail, qmail-inject, qmail-smtpd, qmail-qmtpd, qmail-qmqpd, and qmail-queue.

Local Receiving Modules

Messages injected locally usually come in through `qmail-inject` or the `sendmail` wrapper. It's also possible to inject messages using `qmail-queue` directly, but this is uncommon.

sendmail

The `sendmail` command is primarily a wrapper around `qmail-inject`. It accepts many of the Sendmail version's options and arguments, translates them to their `qmail-inject` equivalents, ignores irrelevant options, and runs `qmail-inject`.

qmail-inject

`qmail-inject`'s job is ensuring that messages have RFC 2822-compliant headers before passing them on to `qmail-queue`. Chapter 4, "Using qmail," details how environment variables can adjust the appearance of messages passed through `qmail-inject`. The `qmail-header man` page details `qmail-inject`'s header manipulation.

Address Qualification

For e-mail addresses listed in From, To, Cc, and other fields, `qmail-inject` ensures that they're in the format *`localpart@qualifieddomain`*.

- If an address consists only of a *`localpart`*, it appends *`@defaulthost`*.
- If an address consists only of *`localpart@hostname`*, it appends *`.defaultdomain`*.
- If an address looks like *`localpart@hostname+`*, it replaces the + with *`.plusdomain`*.

Recipients

If no recipients are specified on the command line, `qmail-inject` looks for recipients in the To, Cc, Bcc, Apparently-To, Resent-To, Resent-Cc, and Resent-Bcc fields. All Bcc and Resent-Bcc fields are stripped from the header.

Because RFC 2822 requires that all messages have a To or CC field, `qmail-inject` adds one, if necessary, containing this:

```
Cc: recipient list not shown: ;
```

which is an empty *address group*.

Required Fields

`qmail-inject` adds the following fields if they're not provided:

- From—the name of the user who invoked `qmail-inject`.
- Date—the current time in Greenwich Mean Time (GMT).
- Message-Id—not strictly required by RFC 2822, but handy for tracking messages. The value is <*timestamp*`.`*pid*`.qmail@`*qualifiedhostname*>, by default, where *qualifiedhostname* is constructed from *defaulthost* and *defaultdomain*.

Other Features

`qmail-inject` also does the following:

- Resent- header fields—Resent fields are handled similarly to original fields: Resent-Cc is added if Resent-To and Resent-Cc are missing, and Resent-From and Resent-Date are added if necessary.
- Addresses listed in header fields must be separated by commas, so if `qmail-inject` sees addresses separated by spaces, it inserts commas. For example, this field:

```
To: carol david
```

will be rewritten as this:

```
To: carol, david
```

- Return-Path and Content-Length fields are stripped.

Remote Receiving Modules

Messages received remotely usually come in through qmail-smtpd, qmail-qmtpd, or qmail-qmpd, depending upon the protocol used.

qmail-smtpd

qmail-smtpd conducts an SMTP session on standard input and standard output, accepts one or more messages, and passes them on to qmail-queue. qmail-smtpd does *not* handle accepting network connections itself: It must be run by a network server such as tcpserver from the ucspi-tcp package (see Appendix B, "Related Packages"), inetd, or xinetd.

qmail-smtpd expects the environment variables listed in Table A-1 to be set.

Table A-1. TCP Environment Variables

VARIABLE	CONTENTS
PROTO	Always TCP
TCPLOCALHOST	Lowercased domain name of the local host, if available, or unset
TCPLOCALIP	Internet Protocol (IP) address of the local host in dotted-decimal form (x.x.x.x)
TCPLOCALPORT	Local port number of the SMTP session, in decimal
TCPREMOTEHOST	Lowercased domain name of the remote host, if available, or unset
TCPREMOTEINFO	A value, often a username, supplied by the remote host, obtained using the IDENT protocol (RFC 1413), if available, or unset
TCPREMOTEIP	IP address of the remote host in dotted-decimal form
TCPREMOTEPORT	Remote port number of the SMTP session, in decimal

qmail-smtpd begins by displaying a banner message like this:

```
220 dolphin.example.com ESMTP
```

The 220 is a status code that indicates "no problems." See Appendix C, "An Internet Mail Primer," for more information about Simple Mail Transfer Protocol (SMTP) and its status codes. dolphin.example.com is the local host name, and ESMTP advertises that qmail-smtpd implements some SMTP extensions.

After displaying the banner message, qmail-smtpd reads SMTP commands from the client via standard input and outputs its response to standard output. SMTP requires commands to be issued in a certain order and syntax, and qmail-smtpd enforces these restrictions. It also enforces other restrictions based on environment

variables such as RELAYCLIENT and DATABYTES and control files such as badmailfrom, databytes, and rcpthosts. (See Chapter 5, "Managing qmail," for more information about these variables and control files.)

One of the most important checks qmail-smtpd performs is the validation of recipients. Recipients are specified in SMTP RCPT commands, which look like this:

```
RCPT TO:<localpart@domain>
```

If the RELAYCLIENT environment variable is set—meaning that the client is allowed to relay—the value of the variable is appended to *domain*, and the modified recipient is added to the list. RELAYCLIENT is usually set to the empty string, so the recipient address is not modified. However, if RELAYCLIENT is inadvertently set to something such as a single space character, this results in making all injections from relay-approved clients fail because of invalid domain names.

NOTE *There* are *legitimate uses of a non-empty* RELAYCLIENT. *The online qmail FAQ gives an example that uses* RELAYCLIENT *and* control/virtualdomains *to implement envelope rewriting* (http://cr.yp.to/qmail/faq/servers.html#network-rewriting).

If RELAYCLIENT is not set, qmail-smtpd looks in control/rcpthosts and control/morercpthosts.cdb for *domain*. If the domain isn't listed, the recipient is rejected with this message:

```
553 sorry, that domain isn't in my list of allowed rcpthosts (#5.7.1)
```

Accepted messages are passed to qmail-queue to be placed in the queue. If successful, qmail-smtpd reports success, accepting responsibility for delivering the message or returning a bounce message to the sender if it's not deliverable.

Note that qmail-smtpd converts SMTP carriage return/linefeed (CR-LF) newlines into Unix LF newlines, and returns a temporary error when it detects a bare LF—one not preceded by a CR.

qmail-smtpd adds a Received field to each message. This field looks like this:

```
Received: from unknown (HELO dolphin.example.com) (127.0.0.1)
  by 0 with SMTP; 8 Aug 2001 16:02:00 -0000
```

where:

- unknown means that TCPREMOTEHOST was not set.

- dolphin.example.com was the parameter supplied by the client with the HELO command.
- 127.0.0.1 was the value of TCPREMOTEIP.
- 0 was the value of TCPLOCALHOST (in this case, set using the -l option to tcpserver).
- SMTP is the protocol.
- 8 Aug 2001 16:02:00 -0000 is the time and date at which qmail-smtpd received the message.

qmail-qmtpd

qmail-qmtpd does pretty much the same thing that qmail-smtpd does. The difference is that it uses the Quick Mail Transfer Protocol (QMTP) instead of SMTP, which changes the commands and responses. qmail-qmtpd requires the same TCP environment variables listed in Table A-1 and honors the RELAYCLIENT and DATABYTES environment variables and rcpthosts, morercpthosts.cdb, and databytes control files.

qmail-qmqpd

qmail-qmqpd works much like qmail-qmtpd except it does not do any relay control: All recipients are accepted unconditionally. Again, it requires the TCP environment variables listed in Table A-1.

qmail-queue

qmail-queue's job is to accept messages and place them into the queue. It reads a single message from file descriptor zero and an envelope from file descriptor one. Chapter 4, "Using qmail," explains the format of the envelope and qmail-queue's exit status codes.

qmail-queue expects the envelope recipient addresses to be fully specified, including a local part (username, alias, extension address), an @, and a fully qualified domain name.

qmail-queue adds a Received field to the message that looks like this:

```
Received: (qmail 16707 invoked from network); 8 Aug 2001 16:02:00 -0000
```

where:

- 1607 is qmail-queue's process ID.
- invoked from network means qmail-queue was invoked by user qmaild.
- 8 Aug 2001 16:02:00 -0000 is the time and date at which qmail-queue processed the message.

The invoked from phrase may also indicate that qmail-queue was invoked by the user alias via qmail-local (invoked by alias) or user qmails via qmail-send (invoked for bounce).

How Messages Are Placed in the Queue

To guarantee reliability, placing messages in the queue is done in four stages:

1. A file is created in /var/qmail/queue/pid named after qmail-queue's process ID. The file system assigns the file an inode number guaranteed to be unique on that file system. This is qmail's queue ID for the message.

2. The pid/*pid* file is renamed to mess/*split*/*inode*, and the message is written to the file.

NOTE split *is the remainder left from dividing* inode *by the compile-time configuration setting* conf-split. *For example, if* inode *is 95 and* conf-split *is the default, 23, then* split *is 3 (95 divided by 23 is 4 with a remainder of 3.) Computer scientists call this operation* modulo.

3. The file intd/*inode* is created and the envelope is written to it.

4. intd/*inode* is linked to todo/*inode*.

At the moment todo/*inode* is created, the message has been queued. qmail-send eventually (within 25 minutes) notices the new message, but to speed things up, qmail-queue writes a single byte to lock/trigger, a named pipe that qmail-send watches. When trigger contains readable data, qmail-send is awakened, empties trigger, and scans the todo directory.

qmail-queue kills itself if it hasn't successfully completed queuing the message after 24 hours.

Sending Modules

Now we'll look at the modules comprising the sending function: qmail-send, qmail-lspawn, qmail-rspawn, qmail-local, and qmail-remote.

qmail-send

qmail-send is the heart of qmail. It processes messages in the queue and dispatches them to qmail-rspawn and qmail-lspawn. Chapter 5, "Managing qmail," covers qmail-send, but we'll examine qmail-send's functions in the order that a message in the queue would experience them: preprocessing, delivery, and cleanup.

Preprocessing

Preprocessing, like queuing, is done in stages:

1. Upon discovering todo/*inode*, qmail-send deletes info/*split*/*inode*, local/*split*/*inode*, and remote/*split*/*inode*, if they exist.

2. A new info/*split*/*inode* is created, containing the envelope sender address.

3. If the message has local recipients, they're added to local/*split*/*inode*.

4. If the message has remote recipients, they're added to remote/*split*/*inode*.

5. intd/*inode* and todo/*inode* are deleted.

At the moment `todo/`*`inode`* is deleted, the message is considered *preprocessed*.

Recipients are considered local if the domain is listed in `control/locals` or the entire recipient or domain is listed in `control/virtualdomains`. If the recipient is virtual, the local part of the address is rewritten as specified in `virtualdomains`.

`Delivery`

Initially, all recipients in `local/`*`split/inode`* and `remote/`*`split/inode`* are marked *not done*, meaning that `qmail-send` should attempt to deliver to them. On its own schedule, `qmail-send` sends delivery commands to `qmail-lspawn` and `qmail-rspawn` using channels set up by `qmail-start`. When it receives responses from `qmail-lspawn` or `qmail-rspawn` that indicate successful delivery or permanent error, `qmail-send` changes their status in `local/`*`split/inode`* or `remote/`*`split/inode`* to *done*, meaning that it should not attempt further deliveries. When `qmail-send` receives a permanent error, it also records that in `bounce/`*`split/inode`*.

Bounce messages are also handled on `qmail-send`'s schedule. Bounces are handled by injecting a bounce message based on `mess/`*`split/inode`* and `bounce/`*`split/inode`*, and deleting `bounce/`*`split/inode`*.

When all of the recipients in `local/`*`split/inode`* or `remote/`*`split/inode`* are marked *done*, the respective `local` or `remote` file is removed.

Retry Schedules

`qmail-send` uses a simple formula to determine the times at which messages in the queue are retried. If *attempts* is the number of failed delivery attempts so far, and *birth* is the time at which a message entered the queue (determined from the creation time of the `queue/info` file), then:

$$nextretry = birth + (attempts \times c)^{2}$$

where c is a retry factor equal to 10 for local deliveries and 20 for remote deliveries. Table A-2 shows the complete retry schedule for a remote message that's never successfully delivered, with the default *queuelifetime* of 604,800 seconds.

Table A-2. Remote Message Retry Schedule

DELIVERY ATTEMPT	SECONDS	DAY-HOUR:MIN:SEC
1	0	0-00:00:00
2	400	0-00:06:40
3	1,600	0-00:26:40
4	3,600	0-01:00:00
5	6,400	0-01:46:40
6	10,000	0-02:46:40
7	14,400	0-04:00:00
8	19,600	0-05:26:40
9	25,600	0-07:06:40
10	32,400	0-09:00:00
11	40,000	0-11:06:40
12	48,400	0-13:26:40
13	57,600	0-16:00:00
14	67,600	0-18:46:40
15	78,400	0-21:46:40
16	90,000	1-01:00:00
17	102,400	1-04:26:40
18	115,600	1-08:06:40
19	129,600	1-12:00:00
20	144,400	1-16:06:40
21	160,000	1-20:26:40
22	176,400	2-01:00:00
23	193,600	2-05:46:40
24	211,600	2-10:46:40
25	230,400	2-16:00:00
26	250,000	2-21:26:40
27	270,400	3-03:06:40
28	291,600	3-09:00:00
29	313,600	3-15:06:40
30	336,400	3-21:26:40
31	360,000	4-04:00:00
32	384,400	4-10:46:40
33	409,600	4-17:46:40

Table A-2. Remote Message Retry Schedule (continued)

DELIVERY ATTEMPT	SECONDS	DAY-HOUR:MIN:SEC
34	435,600	5-01:00:00
35	462,400	5-08:26:40
36	490,000	5-16:06:40
37	518,400	6-00:00:00
38	547,600	6-08:06:40
39	577,600	6-16:26:40
40	608,400	7-01:00:00

The local message retry schedule is similar, but because of the lower *c*, messages are retried twice as often.

Cleanup

When both `local/`***`split`***`/`***`inode`*** and `remote/`***`split`***`/`***`inode`*** have been removed, the message is dequeued by:

1. Processing `bounce/`***`split`***`/`***`inode`***, if it exists.
2. Deleting `info/`***`split`***`/`***`inode`***.
3. Deleting `mess/`***`split`***`/`***`inode`***.

Partially queued and partially dequeued messages left when a system crash interrupts `qmail-queue` or `qmail-send` are deleted by `qmail-send` using `qmail-clean`, another long-running daemon started by `qmail-start`. Messages with a `mess/`***`split`***`/`***`inode`*** file and possibly an `intd/`***`inode`***—but no `todo`, `info`, `local`, `remote`, or `bounce`—are safe to delete after 36 hours because `qmail-queue` kills itself after 24 hours. Similarly, files in the `pid` directory more than 36 hours old are also deleted.

Local Sending Modules

Messages to be delivered locally are passed from `qmail-send` to `qmail-lspawn`, which invokes `qmail-local` to perform the delivery.

qmail-lspawn

qmail-lspawn reads delivery commands from qmail-send on file descriptor 0 (zero), invokes qmail-local to deliver the messages, and reports the results to qmail-send on descriptor 1 (one).

Before invoking qmail-local, qmail-lspawn determines which local user controls the address so qmail-local can be started with the necessary user ID and group ID. qmail-lspawn first checks the qmail-users database, users/cdb. If the address is not listed there, it runs qmail-getpw. If qmail-getpw doesn't find a matching user, it gives control of the address to the alias user.

qmail-local

qmail-local accepts a message on standard input with envelope information, delivery location, and default delivery instructions supplied as arguments. Before attempting delivery, it constructs a Delivered-To field based on the envelope and checks the message for an identical Delivered-To field. If it finds one, it bounces the message to prevent a mail loop. Chapter 4, "Using qmail," details the actual delivery process. If the delivery is successful, qmail-local returns an exit status of 0 (zero). All other codes indicate either permanent or temporary failure.

Remote Sending Modules

Messages to be delivered remotely are passed from qmail-send to qmail-rspawn, which invokes qmail-remote to perform the delivery.

qmail-rspawn

qmail-rspawn reads delivery commands from qmail-send on file descriptor 0 (zero), invokes qmail-remote to deliver the messages, and reports the results to qmail-send on descriptor 1 (one).

qmail-remote

qmail-remote accepts a message on standard input with envelope information supplied as arguments. After attempting to deliver the message remotely via SMTP, it summarizes its results via reports printed to standard output. Chapter 5, "Managing qmail," explains the format of these reports.

The remote host is specified as one of `qmail-remote`'s arguments. It can be either a fully qualified domain name or an IP address. If it's a domain name, `qmail-remote` checks the Domain Name System (DNS) for a mail exchanger (MX) record for that domain. If the remote host is listed in `control/smtproutes`, `qmail-remote` uses the host specified in `smtproutes`.

APPENDIX B

Related Packages

qmail is a complete Mail Transfer Agent (MTA), but many packages were either designed specifically to add new functionality to qmail or simply work well with qmail. This appendix lists some of these packages, describes them briefly, and provides links for more information.

The unofficial qmail home page, `http://www.qmail.org/`, is the definitive collection of information about qmail-related packages.

checkpassword

checkpassword is the authentication package used by `qmail-pop3d` (really `qmail-popup`). The standard checkpassword package by Daniel Bernstein (`http://cr.yp.to/checkpwd.html`) authenticates users with the Unix password file.

Many alternative checkpassword implementations support other authentication mechanisms including Lightweight Directory Access Protocol (LDAP), Structured Query Language (SQL) databases, Pluggable Authentication Modules (PAM), Authenticated POP (`APOP`), and dedicated POP password files. The unofficial qmail home page contains a checkpassword section with links to these alternative implementations (`http://www.qmail.org/top.html#checkpassword`).

TIP *To verify that your checkpassword program is authenticating properly, you can test it from the command line using Perl. For example, if user* `paul`*'s password is* `Lauren&Natalie`*, the command*

```
perl -e 'printf "%s\0%s\099\0", "paul",
"Lauren&Natalie"' | /bin/checkpassword echo OK 3<&0
```

will output "OK" if the authentication succeeds and nothing if it fails.

Courier-IMAP

Courier-IMAP is an Internet Mail Access Protocol (IMAP) server often used with qmail because it supports maildir mailboxes. Chapter 10, "Serving Mailboxes," covers installing, configuring, and using Courier-IMAP.

Courier-IMAP was written by Sam Varshavchik, who maintains a Web page for it (`http://www.inter7.com/courierimap/`).

daemontools

The daemontools package contains a set of utilities for controlling and monitoring services. It's highly recommended, especially for busy systems. Key utilities included are:

- `supervise`, which monitors a service and restarts it if it dies
- `svscan`, which monitors a service directory and starts `supervise`
- `svc`, which talks to `supervise` and allows one to stop, pause, or restart a service
- `multilog`, which maintains a log for a service, automatically rotating it to keep it under the configured size
- `setuidgid`, which runs a program for the superuser with a normal user's user and group IDs

Gerrit Pape distributes the documentation for daemontools as `man` pages from `http://innominate.org/~pape/djb/`.

daemontools was written by Daniel Bernstein, who maintains a Web page for it (`http://cr.yp.to/daemontools.html`).

djbdns

djbdns is a Domain Name System (DNS) server created by qmail's author, Daniel Bernstein. Like qmail, it provides a secure, reliable, efficient, and modular alternative to the *de facto* standard, which in this case is Berkeley Internet Name Daemon (BIND).

If you're running BIND now—either for providing authoritative name service for your domain(s) or as a caching-only server for improving lookup performance—you should consider switching to djbdns for the same reasons that you're running qmail.

Even if you're not running a name server, you should install djbdns and run `dnscache` to enhance DNS lookup performance for all applications—not just qmail—and reduce outgoing DNS traffic.

The official djbdns Web site is `http://cr.yp.to/djbdns.html`. The unofficial djbdns Web site (`http://www.djbdns.org/`) is another valuable resource, as is "Life with djbdns," a djbdns manual available on the Web (`http://www.lifewithdjbdns.org/`).

dot-forward

Sendmail uses `.forward` files to allow users to control the delivery of messages they receive. qmail uses a similar mechanism: `.qmail` files. The dot-forward package gives qmail the ability to use `.forward` files. Systems running Sendmail or any other MTA that uses `.forward` files might want to consider using dot-forward to avoid having to convert existing `.forward` files to their `.qmail` equivalents—or simply to make the transition to qmail less visible to their users.

dot-forward is a small package, so it's easy to install and configure. Chapter 7, "qmail Configuration: Advanced Options," covers installing and configuring dot-forward.

dot-forward was written by Daniel Bernstein, who maintains a Web page for it (http://cr.yp.to/dor-forward.html).

ezmlm

ezmlm is a high-performance, easy-to-use mailing-list manager (MLM) for qmail. If you're familiar with LISTSERV or Majordomo, you know what a mailing-list manager does. For more information about mailing lists under qmail, including installing, configuring, and using ezmlm, see Chapter 9, "Managing Mailing Lists."

ezmlm was written by Daniel Bernstein, who maintains a Web page for it (`http://cr.yp.to/ezmlm.html`).

ezmlm-idx

ezmlm-idx is an add-on for ezmlm that adds many useful features such as multiple message archive retrieval, digests, message and subscription moderation, and

remote list maintenance. Chapter 9, "Managing Mailing Lists," covers installing, configuring, and using ezmlm-idx.

Fred Lindberg and Fred B. Ringel created ezmlm-idx and maintain a Web page for it (`http://www.ezmlm.org/`).

fastforward

fastforward is another Sendmail compatibility add-on. Sendmail uses a central alias database kept in a single file, usually `/etc/aliases`. qmail uses a series of dot-qmail files in `/var/qmail/alias`, one file per alias. If you're migrating to qmail and you've got a Sendmail-format aliases file that you don't want to convert, fastforward gives qmail the ability to use the `aliases` file as-is.

Chapter 7, "qmail Configuration: Advanced Options," covers installing and configuring fastforward. fastforward was written by Daniel Bernstein, who maintains a Web page for it (`http://cr.yp.to/fastforward.html`).

getmail

getmail is a POP3 client written in Python. It retrieves messages from a Post Office Protocol version 3 (POP3) server and delivers them locally to maildir or mbox mailboxes or programs.

Chapter 10, "Serving Mailboxes," covers installing, configuring, and using getmail.

getmail was written by Charles Cazabon, who maintains a Web page for it (`http://www.qcc.sk.ca/~charlesc/software/getmail-2.0/`).

maildrop

maildrop is a mail filter similar to Procmail. It provides a powerful filtering language and can be used in dot-qmail files to intercept junk mail or direct mail to different mailboxes.

maildrop was written by Sam Varshavchik, who maintains a Web page for it (`http://www.flounder.net/~mrsam/maildrop`).

mess822

mess822 is a library and set of applications for parsing Request For Comments (RFC) 822 (currently RFC 2822)–compliant mail messages. The applications include:

- `ofmipd`, a Simple Mail Transfer Protocol (SMTP) daemon that accepts messages from clients and rewrites From fields based on a database
- `new-inject`, a `qmail-inject` replacement that supports user-controlled host name rewriting
- `iftocc`, a dot-qmail utility for checking whether a message was sent to a specific address
- `822header`, `822field`, `822date`, and `822received`, which extract information from a message
- `822print`, pretty-prints a message

mess822 was written by Daniel Bernstein, who maintains a Web page for it (`http://cr.yp.to/mess822.html`).

oMail-webmail

oMail-webmail is a Web-based Mail User Agent (MUA) for qmail. It accesses maildir mailboxes directly, rather than through POP3 or IMAP like some other Web-based MUAs.

oMail-webmail was written by Olivier Müller, who maintains a Web page for it (`http://webmail.omnis.ch/omail.pl?action=about`).

oSpam

oSpam is an anti-spam utility similar to TMDA (see "TMDA" in this appendix).

oSpam was written by Olivier Müller, who maintains a Web page for it (`http://omail.omnis.ch/ospam/`).

qlogtools

qlogtools is a set of utilities used for producing and analyzing logs from services managed using daemontools, especially qmail. Particularly handy are:

- `qlogselect`, which extracts messages from a `multilog qmail-send` log file from a particular sender or time period
- `tai64n2tai`, which converts `multilog` timestamps to the format that qmailanalog requires (see "qmailanalog" in this appendix)
- `multitail`, which displays data appended to a named file, even if the file is cycled—like `multilog`'s `current`

qlogtools was written by Bruce Guenter, who maintains a Web page for it (`http://untroubled.org/qlogtools/`).

qmail-autoresponder

Autoresponders are utilities run from dot-qmail files that send a message in response to incoming mail. Although this sounds like a simple task, there are many pitfalls. For example, you probably don't want to automatically respond to messages received from mailing lists. And you certainly don't want to respond to other autoresponders, which could quickly result in thousands of messages bouncing back and forth between the two responders.

Bruce Guenter's qmail-autoresponder avoids these pitfalls and is easy to install and use. It's available on the Web (`http://untroubled.org/qmail-autoresponder/`).

qmail-qfilter

qmail-qfilter is a utility that allows messages to be filtered before they're passed to `qmail-queue`. Using qmail-qfilter, programs and scripts can be used to alter messages (add or adjust header fields, for example) or refuse unwanted messages (such as those with executable attachments).

Bruce Guenter wrote qmail-qfilter and maintains a Web page for it (`http://untroubled.org/qmail-qfilter/`).

Qmail-Scanner

Qmail-Scanner is a virus-scanning harness for qmail. It works with most commercial Unix virus scanners and includes a built-in scanner that allows administrators to quickly and easily block Windows-based malware before it's been added to the commercial scanner's signature database. Chapter 12, "Understanding Advanced Issues," covers installing, configuring, and using Qmail-Scanner.

Qmail-Scanner was written by Jason Haar, who maintains a Web page for it (`http://qmail-scanner.sourceforge.net/`).

qmail-vacation

qmail-vacation is a special-purpose autoresponder (see the "qmail-autoresponder" section in this appendix) for notifying senders that the recipient won't be reading their message immediately, for example, because they're on vacation. qmail-vacation allows users to easily enable and disable these autoresponses.

qmail-vacation was written by Peter Samuel and is available from the Web (`http://www.gormand.com.au/peters/tools/`).

qmailanalog

qmailanalog processes `qmail-send`'s log files and produces a series of reports that tell you how much and what kind of work the system is doing. If you need statistics about how many messages are being sent or received, how big they are, and how quickly they're being processed, qmailanalog is what you need.

qmailanalog relies on log-entry timestamps in the fractional second format used by `accustamp`, an obsolete time-stamping utility. To use it with logs generated by `multilog`, which are in Temps Atomique International 64-bit, nanosecond precision (TAI64N) format, you'll need to translate them into the old format. One program to do that is available from `http://www.qmail.org/tai64nfrac`.

qmailanalog was written by Daniel Bernstein, who maintains a Web page for it (`http://cr.yp.to/qmailanalog.html`).

Using qmailanalog

Installing qmailanalog following the directions in INSTALL is straightforward, but running it is a little tricky, especially because it doesn't understand TAI64N timestamps:

1. First, install qmailanalog using the directions in INSTALL.

2. Download tai64nfrac to /usr/local/src/tai64nfrac.c, edit the file and remove everything above the line containing /* Id. Compile the program using:

```
# cc -o /usr/local/bin/tai64nfrac /usr/local/src/tai64nfrac.c
#
```

3. Process one or more qmail-send log files through tai64nfrac. The input log entries should look something like this:

```
@400000003b88ef5c313ae1e4 status: local 0/10 remote 0/20
```

   In particular, each line must start with a TAI64N timestamp (the @400000003b88ef5c313ae1e4 part) followed by a qmail-send log message (the status: local 0/10 remote 0/20 part).

   For example:

```
# tai64nfrac < /var/log/qmail/@* > logs.frac
#
```

   The lines in logs.frac should look like this:

```
998829906.825942500  status: local 0/10 remote 0/20
```

4. Process the fractional-timestamp log file using matchup. Because the log files may contain entries for messages that haven't been delivered yet, matchup will write those entries to file descriptor 5. Save them to a file for inclusion in the next matchup run. For example:

```
# /usr/local/qmailanalog/bin/matchup < logs.frac > logs.match 5> logs.cont
#
```

   Each line in logs.match contains all of the relevant information for a single delivery attempt.

5. Use the scripts in /usr/local/qmailanalog/bin with names starting with z to produce a report from logs.match. For example, to produce an overall summary report:

```
/usr/local/qmailanalog/bin/zoverall < logs.match
```

   Each report includes an explanation of its output.

6. Use the scripts in /usr/local/qmailanalog/bin with names starting with x to extract entries for particular messages, senders, or recipients. These extracted entries can then be passed through a z script. For example, for a report on the recipient hosts of messages from root@dolphin.example.com, you could do this:

```
# cd /usr/local/qmailanalog
# bin/xsender root@dolphin.example.com < log.match | bin/zrhosts
. . .report. . .
```

safecat

safecat reliably delivers a message to a maildir mailbox. It's useful for filing messages using filters such as Procmail. For example, the following recipe files all messages in $HOME/Maildir:

```
:0w
|safecat Maildir/tmp Maildir/new
```

safecat was written by Len Budney, who maintains a Web page for it (http://www.pobox.com/~lbudney/linux/software/safecat.html).

serialmail

qmail was designed for systems with full-time, high-speed connectivity. serialmail is a set of tools that help adapt qmail to intermittent, low-speed connectivity. With serialmail on such a system, qmail can be configured to deliver all remote mail to a single spool maildir. The serialmail maildirsmtp command can be used to upload the maildir to the Internet Service Provider's (ISPs) mail hub after the connection is brought up. If the ISP supports QMTP (see Chapter 7, "Configuring qmail: Advanced"), maildirqmtp can also be used.

serialmail can be used on the ISP side of a dialup connection to implement the AutoTURN mechanism, where an SMTP connection by a client causes the

server to initiate a connection back to the client for delivering messages queued on the server. This is similar to the SMTP ETRN function but doesn't require the client to issue the ETRN command. AutoTURN is documented in the AUTOTURN file in the serialmail source directory.

serialmail was written by Daniel Bernstein, who maintains a Web page for it (http://cr.yp.to/serialmail.html).

SqWebMail

SqWebMail is a Web-based MUA like oMail-webmail (see the "oMail-webmail" section in this appendix). SqWebMail also access maildir mailboxes directly, not through POP3 or IMAP, for improved performance.

SqWebMail was written by Sam Varshavchik, who maintains a Web page for it (http://inter7.com/sqwebmail/).

syncdir

The syncdir package for Linux is a library that provides wrapped versions of the link(), open(), rename(), and unlink() system calls that force their changes to be written to disk immediately. This is useful because qmail relies upon the Berkeley Software Distribution (BSD) behavior of these operations to ensure that the queue is crash proof.

To use this library, build and install the library using make and make install, append -lsyncdir to the command in conf-ld in the qmail source directory, and rebuild qmail.

syncdir was written by Bruce Guenter. It's available from the Web (http://untroubled.org/syncdir/).

TMDA

TMDA (Tagged Message Delivery Agent) is an anti-spam tool similar to oSpam (see the "oSpam" section in this appendix). TMDA allows the user to maintain a *whitelist*, a list of pre-approved senders. Messages from all other senders are held until the sender responds to a confirmation request. Because most spammers use invalid return addresses or are too lazy to respond to confirmation requests, their messages are not delivered.

TMDA is covered in Chapter 8, "Controlling Junk Mail."

TMDA was written by Jason Mastaler, who maintains a Web page for it (http://software.libertine.org/tmda/).

ucspi-tcp

qmail's SMTP server doesn't run as a stand-alone daemon. A helper program such as `tcpserver`, `inetd`, or `xinetd` runs as a daemon. When it receives a Transmission Control Protocol (TCP) connection to port 25, the SMTP port, it runs `qmail-smtpd`.

Inetd is the de facto standard network server "super-server." It can be configured through `/etc/inetd.conf` to run `qmail-smtpd`, but the recommended server tool for qmail is `tcpserver`, which is part of the ucspi-tcp package. ucspi-tcp is an acronym for Unix Client-Server Program Interface for TCP, and it's pronounced *ooks-pie tee see pee*.

Chapter 2, "Installing qmail," compares `tcpserver` to inetd. Chapter 8, "Controlling Junk Mail," covers installing and configuring `rblsmtpd`, an ucspi-tcp tool for checking DNS blacklists.

Gerrit Pape distributes the documentation for ucspi-tcp as `man` pages from `http://innominate.org/~pape/djb/`.

ucspi-tcp was written by Daniel Bernstein, who maintains a Web page for it (http://cr.yp.to/ucspi-tcp.html).

VMailMgr

VMailMgr (Virtual Mail Manager) is a virtual domain and virtual user management add-on for qmail. VMailMgr allows a single Unix user, the virtual domain manager, to add and remove mail users in the domain. The virtual users don't require Unix accounts, and can retrieve their mail via POP3 or IMAP.

Chapter 11, "Managing Virtual Domains and Users," covers installing, configuring, and using VMailMgr.

VMailMgr was created by Bruce Guenter, who maintains a Web site for it (`http://www.vmailmgr.org/`).

Vpopmail

Vpopmail is a virtual domain and virtual user management add-on for qmail similar to VMailMgr (see the previous section).

Chapter 11, "Managing Virtual Domains and Users," covers installing, configuring, and using it.

Vpopmail is maintained by Ken Jones, who also maintains a Web site for it (`http://www.inter7.com/vpopmail/`).

APPENDIX C

How Internet Mail Works

Although Internet mail is one of the most heavily used Internet services, many users—and a surprising percentage of system administrators—don't really understand how it works. This appendix provides some background about how Internet mail works and includes pointers to more detailed sources of information.

How a Message Gets from Point A to Point B

When a user on one host sends a message to a user on another host, many things happen behind the scenes of which you may not be aware.

Let's say Alice, `alice@alpha.example.com`, wants to send a message to Bob, `bob@beta.example.com`. Here's what happens:

1. Alice composes the message with her mail user agent (MUA), something such as Mutt or Pine. She specifies the recipient in a To field, the subject of the message in a Subject field, and the text of the message itself. It looks something like this:

```
To: bob@beta
Subject: lunch

How about pizza?
```

2. When she's satisfied with the message, she tells the MUA to send it.

3. At this point, the MUA can add additional header fields such as Date and Message-Id and modify the values Alice entered (for example, it could replace `bob@beta` with `Bob <bob@beta.example.com>`).

4. Next, the MUA *injects* the message into the mail system in one of two ways: It can run a program provided by the mail system for the purpose of injecting messages, or it can open a connection to the Simple Mail Transfer Protocol (SMTP) port on either the local system or a remote mail

server. For this example, we'll assume the MUA uses a local injection program to pass messages to the MTA. The details of the injection process vary by MTA, but on Unix systems the `sendmail` program is a *de facto* standard. With this program, the MUA puts the header and body in a file, separated by a blank line, and passes the file to the `sendmail` program.

5. If the injection succeeds—the message was syntactically correct and `sendmail` was invoked properly—the message is now the MTA's responsibility. Details vary greatly by MTA, but generally the MTA on `alpha` examines the header to determine where to send the message, opens an SMTP connection to `beta`, and forwards the message to the MTA on the `beta` system. The SMTP dialogue requires messages to be sent in two parts: the *envelope*, which specifies the recipient's address (`bob@beta.example.com`) and the return address (`alice@alpha.example.com`), and the message itself, which consists of the header and body.

6. If the `beta` MTA rejects the message, perhaps because there's no user `bob` on the system, the MTA on `alpha` sends a *bounce* message to the return address, `alice@alpha.example.com`, to notify her of the problem.

7. If the `beta` MTA accepts the message, it looks at the recipient's address, determines whether it's local to `beta` or on a remote system. In this case, it's local, so the MTA either delivers the message itself or passes it to a message delivery agent (MDA) like `/bin/mail`, `qmail-local`, or Procmail.

8. If the delivery fails, perhaps because Bob has exceeded his mail quota, the `beta` MTA sends a bounce message to the envelope return address, `alice@alpha.example.com`.

9. If the delivery succeeds, the message waits in Bob's mailbox until his MUA reads it and displays it.

Envelopes vs. Headers

When you send a letter by "snail mail"—the old-fashioned physical delivery method—you write a letter that looks something like this:

```
To: Jane Doe
    123 Main Street
    Springfield, Anystate 99999

Dear Jane,
```

```
Blah blah blah. . .

Your friend, John Q. Public
```

You then place the letter in an envelope with Jane's address, and your address—so the postal service can return your letter to you if can't it deliver it for some reason.

Internet mail works much the same. When you send a letter by e-mail you construct a message that looks like this:

```
From: "John Q. Public" <jqpublic@isp.example.net>
To: "Jane Doe" <jane@doe.example.com>
Subject: Blah

Blah blah blah. . .

-John
```

When you hit the Send button on your MUA, either the MUA or the MTA that receives the message constructs an envelope for it. As with snail mail, the envelope contains the recipient's address—which is required for delivering the message, of course—and the sender's address, which might be necessary for notifying the sender that the letter was undeliverable. Unlike snail mail, the Internet mail envelope does not require a stamp.

Because the envelope is constructed automatically for the user, many users don't even realize it exists. They mistakenly believe that the header of the message *is* the envelope. This belief works fine for simple person-to-person messages such as the one in the previous example, where the envelope is constructed from the header. It fails miserably when that's not the case, such as most spam, messages received from mailing lists, and messages received via Bcc (blind carbon copy).

For example, let's look at message sent with a Bcc header field. The most common implementation of Bcc is for the sending MUA/MTA to strip the Bcc field from the message after adding the addresses listed to the envelope. So, a message that looks like this:

```
From: "John Q. Public" <jqpublic@isp.example.net>
To: "Jane Doe" <jane@doe.example.com>
Bcc: "John Doe" <john@doe.example.com>
Subject: Blah

Blah blah blah. . .

-John
```

when it's submitted by the sender, arrives in both recipients' mailboxes without the Bcc field. This is accomplished by creating an envelope that looks like this:

```
Sender:jqpublic@isp.example.net
Recipients:jane@doe.example.com, john@doe.example.com
```

MTAs have different ways of storing the envelopes of messages in their queues. With qmail, envelope senders are stored under `/var/qmail/queue/info`, local recipients are stored under `/var/qmail/queue/local`, and remote recipients are stored under `/var/qmail/queue/remote`. Messages (header plus body) are stored under `/var/qmail/queue/mess`.

When messages are sent via SMTP, the `MAIL` command is used to send the envelope sender, and the `RCPT` command is used to send envelope recipients. Messages are sent using the `DATA` command, which must be preceded by the `MAIL` and `RCPT` commands.

Finding More Information

For information about how Internet mail works, see one or more of the following documents by the author of qmail:

- Internet mail (`http://cr.yp.to/im.html`)
- SMTP (`http://cr.yp.to/smtp.html`)
- Internet mail message header format (`http://cr.yp.to/immhf.html`)

Detailed information about the Internet mail standard is contained in the Requests for Comment (RFCs).

Internet Mail RFCs

RFCs are the official documentation of the Internet. Most of these are well beyond the commentary stage and actually define Internet protocols such as the Transmission Control Protocol (TCP), the File Transfer Protocol (FTP), Telnet, and the various mail standards and protocols:

- RFC 821, Simple Mail Transfer Protocol, obsoleted by RFC 2821
  `http://www.ietf.org/rfc/rfc0821.txt`
- RFC 822, Standard for the Format of ARPA Internet Text Messages, obsoleted by RFC 2822
  `http://www.ietf.org/rfc/rfc0822.txt`

- RFC 931, Authentication Server
  `http://www.ietf.org/rfc/rfc0931.txt`

- RFC 974, Mail Routing and the Domain System
  `http://www.ietf.org/rfc/rfc0974.txt`

- RFC 1123, Requirements for Internet Hosts—Application and Support
  `http://www.ietf.org/rfc/rfc1123.txt`

- RFC 1413, Identification Protocol
  `http://www.ietf.org/rfc/rfc1413.txt`

- RFC 1423, Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers
  `http://www.ietf.org/rfc/rfc1423.txt`

- RFC 1651, SMTP Service Extensions
  `http://www.ietf.org/rfc/rfc1651.txt`

- RFC 1652, SMTP Service Extension for 8bit-MIMEtransport
  `http://www.ietf.org/rfc/rfc1652.txt`

- RFC 1806, Content-Disposition Header
  `http://www.ietf.org/rfc/rfc1806.txt`

- RFC 1854, SMTP Service Extension for Command Pipelining
  `http://www.ietf.org/rfc/rfc1854.txt`

- RFC 1891, SMTP Service Extension for Delivery Status Notifications
  `http://www.ietf.org/rfc/rfc1891.txt`

- RFC 1892, The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages
  `http://www.ietf.org/rfc/rfc1892.txt`

- RFC 1893, Enhanced Mail System Status Codes
  `http://www.ietf.org/rfc/rfc1893.txt`

- RFC 1894, An Extensible Message Format for Delivery Status Notifications
  `http://www.ietf.org/rfc/rfc1894.txt`

- RFC 1939, Post Office Protocol, Version 3
  `http://www.ietf.org/rfc/rfc1939.txt`

- RFC 1985, SMTP Service Extension for Remote Message Queue Starting (ETRN)
  http://www.ietf.org/rfc/rfc1985.txt

- RFC 1991, PGP Message Exchange Formats
  http://www.ietf.org/rfc/rfc1991.txt

- RFC 2015, MIME Security with Pretty Good Privacy (PGP)
  http://www.ietf.org/rfc/rfc2015.txt

- RFC 2045, MIME Internet Message Bodies
  http://www.ietf.org/rfc/rfc2045.txt

- RFC 2046, MIME Media Types
  http://www.ietf.org/rfc/rfc2046.txt

- RFC 2047, MIME Headers
  http://www.ietf.org/rfc/rfc2047.txt

- RFC 2048, MIME Registration Procedures
  http://www.ietf.org/rfc/rfc2048.txt

- RFC 2049, MIME Conformance Criteria
  http://www.ietf.org/rfc/rfc2049.txt

- RFC 2142, Mailbox Names for Common Services
  http://www.ietf.org/rfc/rfc2142.txt

- RFC 2183, Content Disposition Header
  http://www.ietf.org/rfc/rfc2183.txt

- RFC 2821, Simple Mail Transfer Protocol
  http://www.ietf.org/rfc/rfc2821.txt

- RFC 2822, Internet Message Format
  http://www.ietf.org/rfc/rfc2822.txt

A comprehensive list of mail-related RFCs is available from the Internet Mail Consortium at http://www.imc.org/mail-standards.html.

APPENDIX D

qmail Features

Chapter 1 listed qmail's features in a readable, newbie-friendly format. This appendix does it again in more detail. If you really need to understand a feature, perhaps to explain it to someone else, this appendix should help. This should also help explain the feature list on the official qmail Web site (`http://cr.yp.to/qmail.html`).

Setup Features

qmail includes the following setup features:

- **Adaptable**. During the build process, qmail automatically adapts itself to most Unix and Linux distributions, obviating the need for manual porting by a system programmer.
- **Automatic configuration**. Basic per-host configuration is done automatically by qmail using the `config` and `config-fast` scripts.
- **Quick installation**. Setting up a basic installation is easy and doesn't require lots of decision making.

Security Features

qmail includes the following security features:

- **Compartmentalization of delivery targets**. There is a clear distinction between addresses, files, and programs that prevents attackers from writing to security-critical files and executing arbitrary programs with elevated privileges.
- **Minimization of `setuid()` code**. Only one module, `qmail-queue`, runs `setuid()`.
- **Minimization of root code**. Only two modules runs as `root`: `qmail-start` and `qmail-lspawn`.

- **Five-way trust partitioning**. Five qmail-specific user IDs are used to partition trust within the qmail system. A compromise to the system should be contained to one partition.

- **Logging**. Using the `QUEUE_EXTRA` compile-time option, logging of one-way message hashes, entire message contents, or other desired information is possible for all messages or subsets of messages (for example, messages from or to a specified user or domain).

Message Construction

qmail includes the following message construction features:

- **RFC compliant**. Messages built by `qmail-inject` comply with the Internet RFCs 2822 (message format) and 1123 (requirements for Internet hosts).

- **Address groups**. Full support is provided for RFC 2822 address groups.

- **`sendmail` hook**. A `sendmail` command is included for compatibility with current user agents.

- **Long header fields**. Header line length is limited only by the available system memory.

- **Host masquerading**. Local hosts can be hidden behind a public mail relay.

- **User masquerading**. Local users can be hidden behind aliases on a mail server.

- **Automatic Mail-Followup-To creation**. The Mail-Followup-To header field is used by the author of a message to direct replies to mailing-list messages to the appropriate address or addresses.

SMTP Service

qmail includes the following Simple Mail Transfer Protocol (SMTP) service features:

- **RFC compliant**. Complies with RFC 2821 (SMTP), RFC 1123, RFC 1651 (ESMTP), RFC 1652 (8-bit MIME), and RFC 1854 (pipelining).

- **8-bit clean**. Accepts 7-bit ASCII characters as well as 8-bit extended characters.

- **Supports IDENT (RFC 931/1413/TAP) callback**. This allows cooperating mail administrators to determine the identity of users abusing the system.

- **Relay control**. qmail automatically denies unauthorized relaying by outsiders.

- **Automatic recognition of local Internet Protocol (IP) addresses**. Messages to `jessica@[192.168.1.3]` are recognized as local by `qmail-smtpd` if 192.168.1.3 is a local IP address.

- **Per-buffer timeouts**. Each new buffer of data from the remote SMTP client has its own time limit.

- **Hop counting for detection of looping messages**. Messages that pass through more than 100 delivery hops are rejected.

- **Parallelism limit**. Using `tcpserver`, which is part of the ucspi-tcp package, the number of concurrent incoming SMTP sessions can be controlled.

- **Refusal of connections from known abusers**. Using `tcpserver`, specific hosts and domains can be refused access to the SMTP server.

- **Relaying and message rewriting for authorized clients**. The `RELAYCLIENT` environment variable can be used to allow authorized hosts to relay or to modify header fields for specified hosts.

- **Optional RBL support**. Using `rblsmtpd`, from the ucspi-tcp package, access can be denied to known senders of junk e-mail—also known as *spam*.

Queue Management

qmail includes the following queue management features:

- **Instant handling of messages added to queue**. New messages are always delivered immediately, subject to resource availability.

- **Parallelism limits**. The number of simultaneous local and remote deliveries is limited and configurable.

- **Split queue directory**. Some Unix file systems experience significant slow downs with large directories. qmail splits the queue into a configurable (at compilation) number of subdirectories to keep the number of files per directory low.

- **Quadratic retry schedule**. Undelivered messages in the queue are retried less frequently as they age—the longer a host has been unreachable, the less likely it is to be reachable soon.

- **Independent message retry schedules**. Each message has its own retry schedule. If a long-down host comes back, qmail won't immediately flood it with a huge backlog.

- **Automatic safe queuing**. No mail is lost if the system crashes.

- **Automatic per-recipient checkpointing**. Each successful delivery of a message to multiple recipients is recorded, preventing the sending of duplicates in the event of a crash.

- **Automatic queue cleanups**. Interrupted queue injections can leave partially injected messages in the queue. `qmail-send` automatically cleans these out after 36 hours. `qmail-clean` removes messages after successful delivery.

- **Queue viewing**. `qmail-qread` displays the current contents of the queue.

- **Detailed delivery statistics**. The qmailanalog package analyzes the `qmail-send` logs and produces delivery statistics.

Bounces

qmail includes the following non-deliverability report (bounce) features:

- **`qmail-send` Bounce Message Format (QSBMF) bounce messages**. qmail's bounce messages are in a format that's both machine-readable and human-friendly.

- **Hash Convention for Mail System Status Codes (HCMSSC) support**. Bounce messages include language-independent RFC 1893 error codes.

- **Double bounces sent to postmaster**. Undeliverable bounce messages often indicate configuration errors, so they're delivered to the postmaster alias, by default.

Routing by Domain

qmail includes the following features for routing messages by domain name:

- **Unlimited names for local host**. The local system can have any number of aliases.
- **Unlimited virtual domains**. One host can support any number of virtual domains—each with a separate name space.
- **Domain wildcards**. Using the virtual domains support, domains can be wildcard matched for special routing.
- **Configurable "percent hack" support**. Sendmail-style routed addresses—for example, `molly%example.com@example.net`—can be supported.

SMTP Delivery

qmail includes the following SMTP delivery features:

- **RFC compliant**. Complies with RFC 2821 (SMTP), RFC 974 (Mail Routing), and RFC 1123.
- **8-bit clean**. Sends 7-bit ASCII characters as well as 8-bit extended characters.
- **Automatic downed host backoffs**. If a host is unreachable, qmail waits an hour before trying again.
- **Artificial routing**. Default routes—for example, via DNS MX records—can be overridden using qmail's `smtproutes` configuration file, which is equivalent to Sendmail's `mailertable`.
- **Per-buffer timeouts**. Each new buffer of data to the remote SMTP server has its own time limit.
- **Passive SMTP queue**. Mail can be queued to a mailbox for scheduled delivery using the serialmail package. This is useful for SLIP/PPP.
- **AutoTURN support**. Using the serialmail package, clients can tell the server to send them their queued mail.

Forwarding and Mailing Lists

qmail supports forwarding and mailing lists:

- **Sendmail `.forward` compatibility**. Using the dot-forward package, Sendmail-style `.forward` files can be used.
- **Hashed forwarding databases**. The fastforward package implements a high-performance forwarding database.
- **Sendmail `/etc/aliases` compatibility**. The fastforward package includes a clone of the `newaliases` command that supports Sendmail-style alias databases.
- **Address wildcards**. Using `.qmail-default`, `.qmail-`*`something`*`-default`, and so on, users and mail administrators can specify the disposition of messages to multiple addresses.
- **Mailing-list owners**. If a message is forwarded to `.qmail-`*`something`*, and `.qmail-`*`something`*`-owner` exists, it automatically gets a return address of `user-`*`something`*`-owner@`*`domain`*. This diverts bounces and vacation messages from going to the sender.
- **Variable Envelope Return Path (VERP) support**. VERP allows reliable automatic recipient identification for mailing-list bounces.
- **Delivered-To header field**. Each "final" delivery causes the addition of a Delivered-To header field containing the recipient address. If a message already contains a Delivered-To for the current recipient, the message is rejected. This enables automatic loop prevention, even across hosts.
- **Automatic subscription management**. The ezmlm package allows users to subscribe and unsubscribe themselves from mailing lists. It also tracks bounces and removes invalid addresses.

Local Delivery

qmail supports the following local delivery features:

- **User-controlled address hierarchy**. User `lucy` controls mail addressed to `lucy-`*`anything`*`@`*`domain`*.

- **Supports Unix mbox mailboxes**. Supports the traditional Unix mailbox format: multiple messages in one file, separated by `From` lines.
- **Supports maildir mailboxes**. Provides reliable delivery to mailboxes—even over Network File System (NFS)—using the maildir mailbox format.
- **User-controlled program delivery**. Users can direct messages to filters like Procmail or maildrop, custom scripts, vacation reminders, and so on.
- **Optional new-mail notification**. The `qbiff` program can be used to notify users upon receipt of new messages.
- **Optional Notice-Requested-Upon-Delivery-To (NRUDT) return receipts**. The `qreceipt` program can be used to respond to NRUDTs.
- **Conditional filtering**. The `condredirect` and `bouncesaying` programs can be used to conditionally intercept messages.

POP3 Service

qmail includes a POP3 server with these features:

- **RFC compliant**. Complies with RFC 1939 (POP3).
- **`UIDL` support**. `qmail-pop3d` implements the optional `UIDL` command, which lists the unique ID of one or more messages.
- **`TOP` support**. `qmail-pop3d` implements the optional `TOP` command, which returns the header and beginning of a specified message.
- **Modular password checking**. The checkpassword package, available separately, implements password validation. Versions supporting different authentication methods and user databases are available.
- **Authenticated Post Office Protocol (`APOP`) hook**. `APOP` is available using an alternative checkpassword module.

APPENDIX E

Error Messages

Mail transfer agents (MTAs) are complex systems, and there are thousands of things that go wrong in the process of accepting a message and delivering it locally or remotely. Because qmail was implemented with high reliability as a goal, it's particularly careful in checking for error conditions. Its error messages are generally very descriptive, but sometimes it can be difficult to pinpoint the exact cause of the problem that qmail is complaining about. Rather than attempting to explain each of the hundreds of error messages qmail can generate, this appendix provides some guidance for interpreting these messages.

Error messages can show up in three places: in an interactive shell session, in one of the log files, or in a bounce message.

Interactive Error Messages

Interactive errors are usually the result of "pilot error"—incorrect syntax, permission problems, typographic errors, and so on. First determine whether the error message is coming from the shell or the qmail command you're running. If the message is from a qmail program, consult the `man` page or Chapter 3 (for user commands) or Chapter 5 (for management commands).

If you run across an error message and you can't figure out what the problem is, try searching the archives of the qmail mailing list. Chances are good that someone has already been there, asked that question, and gotten an answer. The list search engine is at `http://www-archive.ornl.gov:8000/`.

Log Messages

The only logs written by qmail are produced by `qmail-send`. (The `qmail-smtpd` logs are really from `tcpserver`). The errors logged by `qmail-send` come from itself, `qmail-local`, or `qmail-remote`.

`qmail-send` *Messages*

Errors logged by `qmail-send` contain the string "alert:" if the problem is critical or the string "warning:" if the problem is serious but not crippling. Errors of either severity are serious and should be investigated immediately.

Critical problems include the inability to access the qmail home directory, the queue, or the control files; the inability to talk to its helpers: `qmail-lspawn`, `qmail-rspawn`, or `qmail-clean`; and the inability to append to a `queue/bounce` file. Verify that the directory it's complaining about exists and has the right owner/group/mode. The easiest way to do this is to stop qmail and do `make check` from the build directory as root.

The message "alert: cannot start: hath the daemon spawn no fire?" means that the communication channels to `qmail-lspawn`, `qmail-rspawn`, or `qmail-clean` weren't set up—perhaps because `qmail-start` had trouble starting them.

qmail-local *Messages*

Errors logged by `qmail-local` are usually temporary and related to permission problems or full file systems (including file systems with space but no free inodes).

Two permanent errors are:

- `This message is looping: it already has my Delivered-To line. (#5.4.6)`. A looping message is a message delivered twice to the same address—usually due to a dot-qmail forwarding instruction that forwards to an address that forwards it back. Check the Received fields and logs to find the culprit.

- `Sorry, no mailbox here by that name. (#5.1.1)`. This is obvious enough, but sometimes you get this message when you're *sure* the mailbox exists. Well, you're probably wrong. The problem is usually because of qmail's narrow definition of a valid mail user, as detailed in the `qmail-getpw man` page and in Chapter 5, "Managing qmail." Typical problems include uppercase characters in the username and home directories not owned by the user.

TIP *See the "RFC 1893 Status Codes" section later in this appendix for an explanation of the "(#x.x.x)" codes.*

Common temporary errors include:

- `Uh-oh: home directory is writable. (#4.7.0)`.

- `Uh-oh: .qmail file is writable. (#4.7.0)`.

Both of these errors indicate that either the user's home directory or dot-qmail file is writable by users that the *conf-patrn* compile-time configuration setting prohibits. (See Chapter 2, "Installing qmail".)

qmail-remote *Messages*

qmail-remote generates its own messages and relays messages from remote hosts. Remote messages are sent by various MTAs but are usually pretty easy to interpret.

Common qmail-remote error messages include:

- Sorry, I wasn't able to establish an SMTP connection. (#4.4.1). qmail-remote has not been able to connect to the remote server. Most likely the remote server is down, but it could also indicate a network problem anywhere between the two systems.

- CNAME lookup failed temporarily. (#4.4.3). qmail-remote tried to look up the Internet Protocol (IP) address of the remote server in the Domain Name System (DNS) and received a temporary error.

- Sorry, I couldn't find any host by that name. (#4.1.2). Again, qmail-remote was unable to find an IP address. This is a temporary error, so it'll keep trying.

- Sorry. Although I'm listed as a best-preference MX or A for that host, it isn't in my control/locals file, so I don't treat it as local. (#5.4.6). qmail-remote looked up the IP address for a remote host and found that it was the local host. However, the host wasn't listed in control/locals or the delivery would have been given to qmail-local. Either fix control/locals or your DNS records.

Bounce Messages

qmail's bounce messages are in a format called QSBMF (qmail-send bounce message format), which is documented on the Web (http://cr.yp.to/proto/qsbmf.txt). RFCs 1892, 1893, and 1894 define another bounce message format called Delivery Status Notification (DSN). The status codes defined in RFC 1893 are also used by QSBMF and other non-DSN bounce message formats. Some MTAs still use their own ad-hoc bounce message formats.

QSBMF

QSBMF messages are designed to be simultaneously human-friendly and easily parsed by automated bounce handlers. Listing E-1 shows a typical bounce message.

Listing E-1. A QSMBF bounce message

```
From: MAILER-DAEMON@dolphin.example.com
To: dave@dolphin.example.com
Subject: failure notice

Hi. This is the qmail-send program at dolphin.example.com.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.

<nosuchuser@dolphin.example.com>:
Sorry, no mailbox here by that name. (#5.1.1)

—- Below this line is a copy of the message.

Return-Path: <dave@dolphin.example.com>
Received: (qmail 3458 invoked by uid 500); 26 Aug 2001 21:56:48 -0000
Date: 26 Aug 2001 21:56:48 -0000
Message-ID: <20010826215648.3457.qmail@dolphin.example.com>
From: dave@dolphin.example.com
to: nosuchuser@dolphin.example.com
```

The body of a QSBMF message consists of four parts: an introductory paragraph, a series of one or more recipient paragraphs, a break paragraph, and a copy of the original message. Blank lines separate the paragraphs.

In this case, the introductory paragraph is:

```
Hi. This is the qmail-send program at dolphin.example.com.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.
```

The initial string, `Hi. This is the. . .`, identifies the message as QSBMF. This paragraph is intended for human readers and identifies the source of the message.

The recipient paragraph in this example is this:

```
<nosuchuser@dolphin.example.com>:
Sorry, no mailbox here by that name. (#5.1.1)
```

The first line identifies the problematic recipient address, and the second line is a description of the problem. The (#5.1.1) is an RFC 1893 status code—these will be explained in the next section.

The break paragraph starts with a – character. In this example, it's this line:

```
—- Below this line is a copy of the message.
```

The remainder of the bounce message is the copy of the original message.

An interesting special case is that of the double bounce: the bounce message sent to the postmaster when a bounce message is undeliverable. With double bounces, the message included is the original bounce message—a QSBMF bounce enclosed in another QSBMF bounce.

RFC 1893 Status Codes

These are three-digit codes displayed as *C.S.D*, where *C* is the *class* sub-code, *S* is the *subject* sub-code, and *D* is the *detail* sub-code.

The class sub-code is one of three values: 2 (success), 4 (temporary error), or 5 (permanent error). They correlate with the initial digits of SMTP status codes.

The subject sub-code has seven possible values, as listed in Table E-1.

Table E-1. RFC 1893 Subject Sub-Codes

CODE	NAME	MEANING
x.0.x	Other or Undefined Status	Problem unknown or undefined
x.1.x	Address Status	Problem with sender or recipient address syntax or validity
x.2.x	Mailbox Status	Problem with the recipient's mailbox
x.3.x	Mail System Status	Problem with the recipient host's mail system
x.4.x	Network and Routing Status	Problem with network or routing
x.5.x	Mail Delivery Protocol Status	Problem with mail delivery
x.6.x	Message Content or Media Status	Problem with message content or format
x.7.x	Security or Policy Status	Problem with security or policy

The detail sub-codes vary with subject sub-code. The only valid detail sub-code for subject sub-code 0 is 0: If an MTA doesn't know what subject sub-code applies, it doesn't make sense to categorize it at a lower level. The detail sub-codes for the other subject sub-codes are listed in Tables E-2 through E-8.

Table E-2. RFC 1893 Address Status Detail Sub-Codes

CODE	MEANING
x.1.0	Unknown problem with an address specified in this message
x.1.1	Nonexistent recipient (part left of @)
x.1.2	Invalid destination host (part right of @)
x.1.3	Bad destination address syntax
x.1.4	Ambiguous destination
x.1.5	Valid destination
x.1.6	Recipient has moved without a forwarding address
x.1.7	Bad sender's address syntax
x.1.8	Bad sender's host

Table E-3. RFC 1893 Mailbox Status Detail Sub-Codes

CODE	MEANING
x.2.0	Unknown problem with an existing mailbox
x.2.1	Mailbox disabled
x.2.2	Mailbox full
x.2.3	Message too big
x.2.4	Problem sending to mailing list

Table E-4. RFC 1893 Mail System Status Detail Sub-Codes

CODE	MEANING
x.3.0	Unknown/other problem with destination host's mail system
x.3.1	Mail system full
x.3.2	Not accepting messages
x.3.3	Mail system doesn't support requested feature
x.3.4	Message too big
x.3.5	Mail system misconfigured

Table E-5. RFC 1893 Network and Routing Status Detail Sub-Codes

CODE	MEANING
x.4.0	Unknown/other network problem
x.4.1	No answer from host
x.4.2	Bad connection
x.4.3	Directory service failure
x.4.4	Unable to route
x.4.5	Mail system congestion
x.4.6	Routing loop detected
x.4.7	Delivery time expired

Table E-6. RFC 1893 Mail Delivery Protocol Status Detail Sub-Codes

CODE	MEANING
x.5.0	Unknown/other problem delivering to next hop
x.5.1	Invalid command
x.5.2	Syntax error
x.5.3	Too many recipients
x.5.4	Invalid command arguments
x.5.5	Wrong protocol version

Table E-7. RFC 1893 Message Content or Message Media Status Detail Sub-Codes

CODE	MEANING
x.6.0	Unknown/other problem with message content
x.6.1	Media (format) not supported
x.6.2	Conversion necessary but prohibited
x.6.3	Conversion necessary but not supported
x.6.4	Message converted but with data loss
x.6.5	Conversion failed

Table E-8. RFC 1893 Security or Policy Status Detail Sub-Codes

CODE	MEANING
x.7.0	Unknown/other security problem
x.7.1	Delivery not authorized, message refused
x.7.2	Delivery to mailing list prohibited
x.7.3	Security conversion required but not possible
x.7.4	Security features not supported
x.7.5	Cryptographic failure
x.7.6	Cryptographic algorithm not supported
x.7.7	Message integrity failure

APPENDIX F

Gotchas

For the most part, qmail works the way people expect it to work. There are, however, a few *gotchas*: quirks in qmail's behavior that frequently cause problems for beginners.

qmail Doesn't Deliver to Superusers

To prevent the possibility of `qmail-local` running commands as a privileged user, qmail ignores all users whose user ID is zero. This is documented in the `qmail-getpw man` page.

That doesn't mean qmail won't deliver to `root`, it just means that such a delivery will have to be handled by a non-privileged user. Typically, one creates an alias for `root` by populating `/var/qmail/alias/.qmail-root` with an entry that forwards to the system administrator's unprivileged account.

qmail Doesn't Deliver to Users Who Don't Own Their Home Directory

This is another security feature and just good general practice. This is documented in the `qmail-getpw man` page.

qmail Doesn't Deliver to Users Whose Usernames Contain Uppercase Letters

qmail converts the entire "local part"—everything before the `@` in an address—to lowercase. The `man` page doesn't come out and say that, but the code does. The fact that it ignores users with uppercase characters is documented in the `qmail-getpw man` page.

qmail Replaces Dots (.) in Extension Addresses with Colons (:)

This is another security feature. The purpose is to prevent extension addresses from backing up the file tree using `..`. Without this restriction, a malicious user could attempt a delivery to `joe-../jane/foo@example.com` hoping to attempt delivery via the dot-qmail file `~joe/.qmail-/../jane/foo`—perhaps disclosing the contents of `~jane/foo` in the form of a bounce message. By replacing dots with colons, qmail ensures that all dot-qmail files for a user are under their home directory. This is documented in the `qmail-local man` page.

qmail Converts Uppercase Characters in Extension Addresses to Lowercase

This is another result of the fact that qmail lowercases the entire local part of addresses, and it is documented in the `qmail-local man` page.

qmail Doesn't Use /etc/hosts

qmail *never* uses `/etc/hosts` to determine the Internet Protocol (IP) address associated with a host name. If you use names in control files, qmail must have access to a name server.

It *is* possible to run qmail on systems without access to a name server. Hosts in control files can be specified by IP address by enclosing them in square brackets ([ ]), for example:

```
[10.1.2.219]
```

Actually, the square brackets aren't always necessary—but it's a good idea to use them anyway.

qmail Doesn't Log SMTP Activity

For whatever reasons, qmail doesn't log SMTP connections, rejections, invalid commands, or valid commands. `tcpserver` can be used to log connections, and `recordio` can be used to log the entire SMTP dialogue. `recordio` is part of the ucspi-tcp package (see Appendix B, "Related Packages"). The procedure is documented in Chapter 7, "Troubleshooting qmail."

qmail Doesn't Generate Deferral Notices

If Sendmail is unable to deliver a message within a few hours, typically four, it sends a deferral notice to the originator. These notices look like bounce messages but don't indicate that the delivery has failed permanently yet.

qmail doesn't send such warnings. A temporarily undeliverable message will only be returned to the originator after it spends at least *queuelifetime* seconds in the queue.

qmail Is Slow If trigger Is Wrong

`qmail-queue` and `qmail-send` communicate via a named pipe called `/var/qmail/queue/lock/trigger`. If this pipe gets messed up, `qmail-send` doesn't notice new messages for up to 25 minutes.

The best way to ensure that `trigger` is set up right is to run `make check` from the source directory. If that's not possible, make sure it looks like this:

```
# ls -l /var/qmail/queue/lock/trigger
prw-w-w-   1 qmails   qmail       0 Jul  5 21:25 /var/qmail/queue/lock/trigger
#
```

Pay particular attention to the `p` at the beginning of the line, which says that it's a named pipe, the mode (must be world-writable), the owner, and the group.

DNS or IDENT Lookups Can Make SMTP Slow

If `qmail-smtpd` is slow to respond to connections, the problem is probably because of Domain Name System (DNS) reverse lookups or IDENT lookups having to time out. If the problem is DNS-related, the best fix is to correct your DNS configuration. Another approach is to configure `tcpserver` not to attempt the lookups. That might be an option if you don't need the information these lookups provide. This can be accomplished by removing the `-h`, `-p`, and `-r` options and adding `-H`, `-P`, `-R`, and `-l 0` (*ell zero*).

qmail-smtpd Accepts Mail for All Recipients

The `control/rcpthosts` file specifies the hosts for which `qmail-smtpd` will accept messages (unless `RELAYCLIENT` is set to allow relaying). However, `qmail-smtpd` does not attempt to validate the recipient. If an invalid recipient is specified, `qmail-send` will generate a bounce message. This is often a problem for simplistic

open-relay testing programs that wrongly assume successful SMTP injection means successful delivery.

For example, a common relay test is to send a message to *recipient%testhost@yourdomain*, which relies on the Sendmail percent hack: stripping *@yourdomain*, replacing the % with @, resulting in *recipient@testhost*, and reinjecting the message to the new address.

With qmail, unless the `control/percenthack` file is in use, such a test merely tries to deliver a message to the local mailbox *recipient%testhost*, which probably doesn't exist. The result is a bounce message sent to the return address specified by the relay tester.

qmail-smtpd Doesn't Automatically Relay for the Local Host

By default, `qmail-smtpd` doesn't accept messages for remote hosts—even if the SMTP client is a Mail User Agent (MUA) running on the local host. If you want to allow local MUAs to inject mail via SMTP, you must enable selective relaying and grant relay access to 127.0.0.1, the local host. See Chapter 3, "Configuring qmail: The Basics," for more information about selective relaying.

Index

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X