# Exim Overview

Date: 9 November 2000

Exim is a mail transfer agent (MTA) developed at the University of Cambridge for use on Unix systems connected to the Internet. It is freely available under the terms of the GNU General Public Licence. In overall style it is similar to Smail 3, but its facilities are more extensive. It contains facilities for verifying incoming sender and recipient addresses, for refusing mail from specified hosts, networks, or senders, and for controlling mail relaying.

Exim is in production use at quite a few sites, some of which move hundreds of thousands of messages per day. This document contains an overview description of the way Exim works, with a certain amount of omission and simplification to keep it fairly short. Please address any enquiries about Exim to Philip Hazel:

Email: <ph10@cus.cam.ac.uk>
Phone: +44 1223 334714
Fax: +44 1223 334679

University of Cambridge
Computing Service
New Museums Site
Pembroke Street
Cambridge CB2 3QG
United Kingdom



---

***If I have seen further it is by standing on the shoulders of giants.*** (Isaac Newton)

## 1. Background

Exim owes a great deal to Smail 3 and its author, Ron Karr. Without the experience of running and working on the Smail 3 code, I could never have contemplated starting to write a new mailer. The general style of operation and configuration are taken from Smail 3, though the actual code of Exim is entirely new.

My intention was to write a mailer that had more functionality than Smail 3, but which retained the simple lightweight approach, as this seemed to me to be all that was needed for systems directly connected to the Internet, where most messages are delivered almost immediately. On the central mail machines at Cambridge University, 'most' means around 98%.

Since the first versions of Exim went into service at the end of 1995, it has continued to develop, and now has far more facilities than my original conception.

## 2. Availability

The current distribution of Exim is available via the Exim page at **http://www.exim.org**. The distribution contains an ASCII copy of the documentation; other formats (HTML, PDF, PostScript, Texinfo) can be downloaded from the web site. The HTML version is also available online.

The following operating systems are currently supported: AIX, BSDI, DGUX, FreeBSD, GNU/Hurd, GNU/Linux, HI-OSF (Hitachi), HP-UX, IRIX, MIPS RISCOS, NetBSD, OpenBSD, QNX, SCO, SCO SVR4.2 (aka UNIX-SV), Solaris (aka SunOS5), SunOS4, Tru64-Unix (aka DEC OSF1, aka Digital UNIX) Ultrix, and Unixware.

## 3. Limitations

For the benefit of those reading this overview to see whether Exim is of interest to them, its limitations are listed first.

- Exim is written in ANSI C. This should not be much of a limitation these days. However, to help with systems that lack a true ANSI C library, Exim avoids making any use of the value returned by the *sprintf()* function, which is one of the main incompatibilities. It has its own version of *strerror()* for use with SunOS4 and any other system that lacks this function, and a macro can be defined to turn *memmove()* into *bcopy()* if necessary.
- Exim is intended for use as an Internet mailer, and therefore handles addresses in RFC 822 domain format only. It cannot handle 'bang paths', though simple two-component bang paths can be converted by a straightforward rewriting configuration. This does not prevent Exim from being interfaced to UUCP, provided domain addresses are used.
- Exim insists that every address it handles has a domain attached. For incoming local messages, domainless addresses are automatically qualified with a configured domain value. Configuration options specify from which remote systems unqualified addresses are acceptable. They are qualified on receipt.
- The only external transport currently implemented is an SMTP transport over a TCP/IP network (using sockets), suitable for machines on the Internet. However, a pipe transport is available, and there are facilities for writing messages to files in 'batched SMTP' format; both of these can be used to send messages to some other transport mechanism such as UUCP. Batched SMTP input is also catered for.
- Exim is not designed for storing mail for dial-in hosts. When the volumes of such mail are large, it is better to get the messages 'delivered' into files (that is, off Exim's queue) and subsequently passed on to the dial-in hosts by other means.

## 4. Main features

Exim follows the same general approach of decentralized control that Smail 3 does. There is no central process doing overall management of mail delivery. However, the independent delivery processes share data in the form of 'hints', which makes delivery more efficient in some cases. The hints are kept in a number of DBM files. If any of these files are lost, the only effect is to change the pattern of delivery attempts and retries.

Here is a summary of Exim's main features. More details are given in the sections which follow.

- Many configuration options can be given as expansion strings, which are transformed in various ways when they are used. As these can include file lookups, much of Exim's operation can be made table-driven if desired. For example, it is possible to do local delivery on a machine on which the users do not have accounts. The ultimate flexibility can be obtained (at a price) by running a Perl interpreter while expanding a string.
- Regular expressions, compatible with Perl 5, are available in a number of configuration parameters.
- Domain lists can include file lookups, making it possible to support a large number of local domains, for example.
- Exim has flexible retry algorithms, applicable to mail routing as well as to delivery.
- Exim contains header and envelope rewriting facilities.
- Unqualified addresses are accepted only from specified hosts or networks.
- Exim can perform multiple deliveries down the same SMTP channel after deliveries to a host have been delayed.
- Exim can be configured to do local deliveries immediately but to leave remote deliveries until the message is picked up by a queue-runner process. This increases the likelihood of multiple messages being sent down a single SMTP connection.

- When copies of a message have to be delivered to more than one remote host, up to a configured maximum number of remote deliveries can be done in parallel.
- Exim has support for the SMTP AUTH extension for authenticating clients.
- Exim has support for encrypted connections using the SMTP STARTTLS extension.
- Exim supports optional checking of incoming return path (sender) and receiver addresses as they are received by SMTP.
- SMTP calls from specific hosts and networks, optionally from specific idents, can be locked out, and incoming SMTP messages from specific senders can also be locked out. Blocked hosts can be identified explicitly, or via RBL lists, or the SMTP AUTH and TLS mechanisms can be used.
- It is possible to control which hosts may use the Exim host as a relay for onward transmission of mail. Again, the SMTP AUTH and TLS mechanisms can be used in this connection.
- Messages on the queue can be ‘frozen’ and ‘thawed’ by the administrator.
- The maximum size of messages can be specified.
- Exim can handle a number of independent local domains on the same machine; each domain can have its own alias files, etc. These are commonly called *virtual domains*.
- Simple mailing lists can be handled directly by Exim itself (but for ‘serious’ mailing list operations, it is best to use it in conjunction with specialist mailing list software).
- Exim stats a user’s home directory before looking for a **.forward** file, in order to detect the case of a missing NFS mount.
- Exim contains an optional built-in mail filtering facility. This enables users to set up their own mail filtering in a straightfoward manner without the need to run an external program. There can also be a system filter file that applies to all messages.
- There is support for multiple user mailboxes controlled by prefixes or suffixes on the user name, either via the filter mechanism or through multiple **.forward** files.
- Periodic warnings are automatically sent to messages’ senders when delivery is delayed – the time between warnings is configurable.
- A queue run can be manually started to deliver just a particular portion of the queue, or those messages with a recipient or sender whose address contains a given string or matches a particular regular expression.
- Exim can be configured to run as root only when needed; in particular, it need not run as root when receiving incoming messages or when sending out messages over SMTP. It always uses *setuid()* when doing local deliveries.
- I have tried to make the wording of delivery failure messages clearer and simpler, for the benefit of those less-experienced people who are now using email. However, the wording can be completely replaced by customized paragraphs supplied in a file if necessary.
- Exim contains support for IPv6.
- The Exim Monitor is an optional extra; it displays information about Exim’s processing in an X window, and an administrator can perform a number of control actions from the window interface. This does not, however, supplant command-line access to all the control functions.

## 5. Performance

Although I did not specifically set out to write a high-performance MTA, Exim does seem to be fairly efficient. One site I heard of was a mailing list exploder that sometimes handles over 100,000 deliveries a day on a big Linux box, the record being 177,000 deliveries (791MB in total). Up to 13,000 deliveries in an hour have been reported. On larger systems, up to 800,000 messages a day have been reported.

## 6. Interface

Like many MTAs, Exim has adopted the Sendmail interface so that it can be a straight replacement for **usr/sbin/sendmail** or **/usr/lib/sendmail**. All the relevant Sendmail options are implemented. There are also some additional options that are compatible with Smail 3, and some further options that are new to Exim.

The runtime configuration interface is a single file which is divided into a number of sections. The entries in this file consist of keywords and values, in the style of Smail 3 configuration files.

Control of messages on the queue can be done via certain privileged command line options. There is also an optional monitor program called **eximon**, which displays current information in an X window and contains interfaces to the command line options.

## 7. Method of operation

When Exim receives a message, it writes two files in its spool directory. The first contains the envelope information, the current status of the message, and the headers, while the second contains the body of the message. The status of the message includes a complete list of recipients and a list of those that have already received the message. The header file gets updated during the course of delivery if necessary.

A message remains in the spool directory until it is completely delivered to its recipients or to an error address, or until it is deleted by an administrator or by the user who originally created it. In cases when delivery cannot proceed – for example, when a message can neither be delivered to its recipients nor returned to its sender, the message is marked ‘frozen’ on the spool, and no more deliveries are attempted. The administrator can thaw such messages when the problem has been corrected, and can also freeze individual messages by hand if necessary.

As delivery proceeds, Exim writes timestamped information about each address to a per-message log file; this includes any delivery error messages. This log is solely for the benefit of the administrator. All the information Exim itself needs for delivery is kept in the header spool file. The message log file is deleted with the spool files. If a message is delayed for more than a configured time, a warning message is sent to the sender.

The main delivery processing elements of Exim are called *directors*, *routers*, and *transports*. Code for a number of these is provided, and compile-time options specify which ones are actually included in the binary. Directors handle addresses that include one of the local domains, routers handle remote addresses, and transports do actual deliveries.

When a message is to be delivered, the sequence of events is roughly as follows:

- If there is a system filter file, it is obeyed. This can check on the contents of the message and its headers, and cause delivery to be abandoned or directed to alternative or additional addresses, or it can freeze the message for human attention. It can also add headers to the message, and set up ‘scores’ which can be used in local recipients’ filter files.
- Each address is parsed and a check is made to see if it is local or not, by comparing the domain with the list of local domains, which can be wildcarded, or even held in a file if there are a large number of them.
- If an address is local, it is passed to each configured director in turn until one is able to handle it. There are directors for handling aliases, for handling **.forward** files, for checking on login names, and so on. If no director can handle it, the address is failed. Directors can be targeted at particular local domains, so several local domains can be processed independently of each other.
- A director that accepts an address may set up a local or a remote transport for it, or it may generate one or more new addresses (typically from alias or forward files). New addresses are fed back into this process from the top, but in order to avoid loops, a director will ignore any address which has an identically-named ancestor that was processed by itself.

- If an address is not local, it is passed to each router in turn until one is able to handle it. There are routers for looking up the domain in a local list, for doing DNS or host file lookups, and for running external routing programs. If no router can handle it, the address is failed.
- A router that accepts an address may set up a transport for it, or may pass an altered address to subsequent routers, or it may discover that the address is a local address after all. This typically happens when a partial domain name is used and (for example) the DNS lookup is configured to try to extend such names. In this case, the address is passed back to the directors.
- Routers normally set up remote transports for messages that are to be delivered to other machines. However, a router can pass a message to a local transport, and by this means messages for remote hosts can be routed to other transport mechanisms such as UUCP via a file or a pipe.
- When all the directing and routing is done, addresses that have been successfully handled are passed to their assigned transports. Local transports normally handle only one address at a time, but remote ones can handle more than one. Each local transport runs in a separate process under a non-privileged uid.
- If there were any errors, a message is returned to an appropriate address (the sender in the common case).
- If one or more addresses suffered a temporary failure, the message is left on the queue, to be tried again later. Otherwise the spool files and message log are deleted.

### 8. Mail filtering

Exim can be configured to allow users to set up filter files as an alternative to the traditional **.forward** files. A filter file can test various characteristics of a message, including the contents of the headers and the start of the body, and direct delivery to specified addresses, files, or pipes according to what it finds. The system-wide filter file uses the same control syntax.

### 9. Directors

The existing directors are listed below. I use the RFC 822 term *local part* to mean that portion of an address that comes before the @ character.

- **aliasfile**: This director handles local part expansion via a traditional alias file. The name of the file is obtained by string expansion, and may therefore depend on the local part or the domain. Generated pipe and file addresses can be (independently) locked out.

  The **aliasfile** director can also be used to test a list of local parts and direct any messages for them to a specific transport. In this case the data associated with the local part in the file is not used for address expansion, but is available for other purposes. For example, files containing records of the form

  ```
  foo: uid=1234 gid=5678 mailbox=/home_1/foo/inbox
  ```

  could be used on a system that did local deliveries without consulting its password file. The **aliasfile** director could use the file to verify that the local part was valid, and then the **appendfile** transport could use it to get a uid, gid, and mailbox for the delivery.

  The **aliasfile** director can also be configured to include the domain in the key that is looked up, making it possible to hold aliases for several different local domains in the same file. It is possible to specify a default address to be used if nothing in the alias file matches the incoming address.

- **forwardfile**: This director handles local part expansion via a traditional forward file or, if so configured, by a user's filter file. The name of the file is obtained by string expansion, and may therefore depend on the local part or the domain, though if it is not an absolute path it is automatically assumed to be in the home directory of the user whose login name is the local part. Mailing lists can be handled by file names of the form

  ```
  /some/list/directory/${local_part}
  ```

and it is possible to specify an error address for each list that depends on the list name. Generated pipe and file addresses can be (independently) locked out.

- **localuser**: This director matches the local part of an address to a user of the machine. It can also be configured to do a pattern match on the user's home directory name. This makes it possible to partition the set of local users according to their home directories.
- **smartuser**: This director matches any local part. It can be used to pass messages for unknown users to a script that generates a helpful error message, or it can be used to send such messages to another host, optionally changing the envelope address in the process.

The configuration file determines which directors are actually used, and in which order. It is possible to use the same director more than once, with different options.

The addresses a director handles can be constrained in the following ways:

- A specific set of local domains may be specified, in which case the director is called only for addresses that contain one of those domains.
- A specific set of local parts may be specified, in which case the director is called only for addresses that contain one of those local parts. This could be used, for example, to handle 'postmaster' independently of the particular local domain.
- A specific set of sender addresses may be specified. This can be used to implement closed mailing lists.
- A director may be configured to handle local parts that start with a certain prefix and/or end with a certain suffix. For example, a director can be set up to handle local parts of the form `xxxx-request` only.
- A flag controls whether a director is called when an address is being verified on an incoming message (during the SMTP dialogue), as opposed to being directed for delivery.

In addition, certain files can be required to exist or not exist for a given director to be run.

**10. Routers**

The existing routers are:

- **domainlist:** This router searches a list of domains for the one it is trying to route. The list may either be a string in the configuration file, possibly including wild cards or regular expressions, or it may be in a file, or both may be provided. In the case of a file, keys of the form `*.foo.bar.com` can be used for simple wildcarding.

  If the domain is found, its entry can either specify a single replacement domain name that is passed on to subsequent routers, or it can specify a list of domain names that are looked up by this router. The lookup can be done by the *gethostbyname()* function, or by DNS lookup, and in the latter case it is configurable whether MX or A records or both are used. As well as providing explicit routing for certain domains, the domainlist router can be used to set up gateways for partial domains (for example, for `*.uucp`) and it can also be used as a 'smarthost' router by using the all-inclusive wild card.
- **lookuphost:** This router looks up domain names either by calling the *gethostbyname()* function, or by using the DNS. In the latter case, it can be configured to use the DNS resolver options for qualifying single-component names and for searching parent domains. It is also possible to specify explicit text strings for widening domains that are not found initially. It is possible to insist on the presence of MX records for certain sets of domains. A configuration option controls whether the message's headers are rewritten when a domain name is changed.
- **queryprogram:** This router passes the address to a command that runs in a separate process under an unprivileged uid and gid. The command returns a line of text specifying whether it matched the domain or not. If it did match, it may specify a transport name, or it may specify that the transport specified for the router is used. The command may also send back a new domain name to replace

the current one, and specify a method of looking this name up (*gethostbyname()*, DNS, or pass to next router).

The configuration file determines which routers are actually used, and in which order. It is possible to use the same router more than once, with different options.

Like directors, routers can be constrained to handle only certain domains or certain local parts (though I haven't seen a good use for that yet), or messages from certain senders. If a router times out, either the delivery can be deferred, or the address can be passed on to the next router.

A flag controls whether a router is called when an address is being verified on an incoming message, as opposed to being routed for delivery.

### 11. Transports

Local and remote transports are handled differently. A local transport is always run in a separate process with an appropriate real uid and gid. Their values can be specified in the transport's configuration, or passed over from the director that handled the address. Remote transports run under Exim's own uid. The existing transports are:

- **appendfile:** This local transport appends the message to a file whose name is specified as a string containing variable expansions. The current local part can be inserted via the expansion mechanism, and file names such as

  ```
  /home/${local_part}/inbox
  /var/mail/${local_part}
  ```

  are typical examples. However, it is possible to look up each individual user's inbox name in a file, should that be required.

  Exclusive access to the file is ensured by using the traditional mailbox locking strategy of creating a lock file. The lock creation process uses a 'hitching post' algorithm (similar to that used by Pine) which is robust when the mailbox file is NFS-mounted. The file is also locked using the *fcntl()* function, but either one of the locking mechanisms can be turned off if necessary.

  The **appendfile** transport can also be configured to deliver each message into a separate file in a given directory, optionally in 'maildir' format. In this case, no locking is needed.

  Options on this transport allow for the insertion of a prefix line (for example, 'From xxx...') and a suffix line, special processing of message lines starting with 'From', and the addition of **Return-path:**, **Delivery-date:**, and **Envelope-to:** headers. If the mailbox file is not a regular file, or does not have the correct owner, group, or permissions, no delivery takes place; the address is deferred and the postmaster is informed, except that, if the file's permissions are *greater* than those required, Exim reduces the permissions and carries on. There are additional checks to reduce the possibility of security exposures caused by race conditions.

- **pipe**: This local transport passes the message via a pipe to a specified command (program or script) which is run in a separate process under a given uid and gid. Various parameters of the message are passed as environment variables, and there are the same options as for **appendfile** for controlling the form of the message.

  The returned status of the command may be used to determine success or failure, or it can be ignored. A configuration option specifies whether any standard output generated by the command is to be returned to the sender (either always, or only if the command fails). If this is set unconditionally and output is actually generated, the delivery is deemed to have failed, whatever the returned status of the command. The maximum amount of output generated by the command can be controlled, and a timeout may be set for it.

- **smtp**: This remote transport delivers a message using SMTP over TCP/IP. All addresses in the message that route to the same set of hosts, and have the same error address (return path), are normally sent in a single transaction. An explicit list of hosts can be set for the transport, or a host list may be attached to an address by one of the routers. If all the hosts are temporarily unable to accept the message, it is delivered to one of a list of fallback hosts, if configured.

## 12. Exim logs

Exim writes to three different log files:

- The main log records the arrival of each message and the result of each delivery attempt in a single line in each case. The format is as compact as possible, in an attempt to keep down the size of log files. A number of other events are also recorded in the main log.
- The reject log records information from messages that are rejected because their return paths are invalid (a configurable option), or the sender or sending host appears in a blocking list. The headers are written to this log if they have been read, following a copy of the one-line message that is also written to the main log.
- The panic log is written to when Exim suffers a disaster and has to bomb out.

A utility script for renaming and compressing the main and reject logs each night is provided. There are also scripts for extracting statistics from log files and for searching log files for message entries that match a given pattern. For example, one can pull out all entries relating to messages for a given local part.

It is possible to configure Exim to write its logging data to syslog instead of, or as well as, to local files.

## 13. Exim databases

Exim maintains a number of databases in DBM files to help it perform efficient mail delivery. In effect, the files contain hints, and if they are lost it is not a disaster – Exim's performance just suffers a bit. The databases are:

- **retry**: This contains information about each failing remote host and temporary failing local delivery – when the first failure was detected, when the delivery (or directing or routing) was last tried, and when it should next be tried. More details about retry algorithms are given below.
- **wait-smtp**: This contains information about messages that are waiting for particular hosts after an SMTP delivery failure (see the next section).
- **reject**: This contains information about SMTP message rejections (see below).
- **serialize-**<*xxx*>: Exim can be configured to serialize connections to certain hosts and some kinds of queue run, in other words, do only one of them at a time. The data for implementing this is kept in database files whose names begin with `serialize-`.

There is a utility program that lists the contents of one of these databases, and another that allows manual modifications to be applied in some cases. Database records are timestamped, and there is a utility that removes records that are older than a given period, and also cleans up **wait-smtp** records containing references to messages that no longer exist. Running this daily or weekly should be sufficient to keep the files reasonably tidy.

Exim can use any one of the following DBM libraries: **ndbm**, **gdbm**, **DB 1.85**, **DB 2.***x*, **DB 3.***x*, or **tdb**.

## 14. SMTP batching

When an SMTP delivery attempt fails, causing the message to be deferred till later, Exim updates a DBM database that contains records keyed by host name plus IP address. Each record holds a list of messages that are waiting for that host and address.

When an SMTP delivery succeeds, Exim consults the database to see if there are any other messages waiting for the same host and address. If it finds any, it creates a new Exim process and passes it the open SMTP channel and a message identification. The new process then delivers the waiting message down the existing channel and may in turn cause the creation of yet another process. Any other waiting addresses in the message are skipped. The maximum number of messages sent down one connection is configurable.

This scheme achieves some SMTP efficiency when a number of messages have been queued up for a given host, without the overhead of a heavyweight queueing apparatus.

## 15. Retries

When a message cannot immediately be directed, routed, or delivered, it remains on the queue and another delivery attempt occurs at a later time. While failures to deliver to remote hosts are the most common cause of this, it is also possible for a message to be deferred as a result of temporary local delivery failure, or following directing or routing. A local delivery can fail if the user is over quota, while directing can be delayed if a user's home directory is not available (for example, a missing NFS mount), and therefore the existence of a **.forward** file cannot be tested. Routing can be delayed by DNS timeouts.

Exim can be given a set of rules which specify how often to retry deferred addresses, and when to give up. These rules apply to directing and routing as well as to transporting, and are keyed by (wildcarded) domain name or, for local users, by local part and domain name, either of which can be wildcarded.

Each rule is actually a sequential list of subrules, which are applied successively as time passes. At present there are two kinds of subrule: fixed interval, and geometrically increasing interval. For example, it is possible to specify a rule such as 'retry every 15 minutes for 2 hours; then increase the interval between retries by a factor of 1.5 each time until 8 hours have passed; then retry every 8 hours until 4 days have passed; then give up'. The times are measured from when the address first failed, so, for example, if a host has been down for two days, new messages will immediately go on to the 8-hour retry schedule.

Exim does not have an elaborate series of alarm clocks to cause retries to happen exactly on schedule. A queue-runner process is started periodically, to attempt delivery, one by one, of messages containing addresses that have passed their next retry time. If such an address fails again, a new retry time is computed, and so subsequent messages queued for the same address get skipped. The queue is not processed sequentially, but in a 'random' order, to prevent one rogue message that causes a problem blocking other messages to the same destination for ever.

When the maximum time for retrying has passed, pending addresses are failed. However, a next try time is still computed from the final subrule. Until that time is reached, any new messages for the address are immediately failed. When the next try time is passed, one further delivery attempt is made; if this fails, a new next try time is computed, and so on.

The increasing number of small computers on the Internet has caused there to be a lot of messages addressed to hosts that are never going to listen. The retry logic described above should reduce the amount of wasted time spent on trying to deliver such messages. However, some administrators are unhappy about this rather draconian approach, which can cause an address to be failed without any deliveries being attempted. Exim can alternatively be configured always to try at least once those hosts whose last failure was before the arrival of the message. This option increases the number of attempts to deliver to dead hosts.

Retry rules can be predicated on particular errors as well as on domain names, and for domains that are looked up in the DNS, further discrimination on whether MX records were used or not is also possible. Thus it is possible to treat 'connection refused' and 'connection timed out' differently, or to distinguish between 'connection refused and there was only an A record' and 'connection refused from a host pointed to by an MX record'.

When a local delivery fails because a user is over quota, the retry rule can be predicated on the length of time since the mailbox was last read. For example, if the mailbox has been recently read, the delivery can be retried for a while; otherwise it can be failed quickly.

## 16. Header rewriting

There are those who argue that header rewriting is a totally Bad Thing; there are others who swear they cannot live without it. Exim provides the facility – you do not have to use it!

Exim can be configured to rewrite the address portions of headers when a message is received. From release 3.20, Exim can also be configured to rewrite addresses in header lines at transport time. Rewriting rules can be targeted at individual headers and the envelope fields; it is possible, for example, just to rewrite the 'From' header and no others.

Rewriting rules are keyed by local part and domain, either of which can be wildcarded, and the replacement text is a general expansion string which can contain file lookups. This makes it possible to replace login names by ‘friendly’ names in outgoing addresses via a DBM lookup, for example. The other most common rewriting requirement of replacing `*.foo.bar` with `foo.bar` is also easily handled.

Headers are also automatically rewritten by Exim in two cases:

- If a locally-generated message contains addresses without domains, a configured qualifying domain is added to each of them. It is also possible to specify which remote systems are permitted to send messages containing unqualified addresses. These too get qualified on reception.
- Routing of a domain may reveal that it was only a partial domain, in which case the headers are rewritten to contain the full domain. For example, as a result of routing, an address such as `xxx@foo` may turn into `xxx@foo.bar.ac.uk`.

In addition to the rewriting rules, Exim can be configured to add or delete specific headers at transport time. The configuration can either be on the transport, in which case it applies to all copies of the message sent by that transport, or on a director or router, in which case it applies only to copies of the message that are associated with the addresses that the router or director handled.

### 17. Host verification

Exim can be configured to accept incoming SMTP calls from certain hosts only, or it can be configured to reject calls from certain hosts. In both cases, the test may include an RFC 1413 identification check. A system that gets all its mail via a central hub might want to lock out the rest of the world, while a number of systems under one management might want to exchange mail only via the standard mailer, and hence reject mail from all but certain specified ids within the group.

Hosts for rejection can be identified explicitly, or via the RBL DNS databases. Alternatively, the SMTP AUTH extension mechanism may be used for authenticating hosts.

When a host fails an acceptance test, Exim can either give an error code immediately on connection, or allow the connection to proceed and then give error codes to all the message’s recipients. The latter approach is useful when using the mechanism to reject unsolicited junk mail and mail bombs, because it normally prevents the sender from trying again with the same message.

### 18. SMTP port reservation

The maximum number of simultaneous incoming SMTP calls can be set, and in addition, a number of them can be reserved for particular hosts or particular IP networks. It is also possible to specify a system load value above which only calls from the reserved hosts are accepted.

### 19. Control of relaying

A host is said to act as a relay if it accepts an incoming message from an external host and delivers it to an external host. Unscrupulous persons have been known to use unsuspecting hosts as relays in an attempt to disguise the origin of messages. An Exim host can be configured to accept mail from any host for onward transmission to a limited set of domains only, and to accept mail only from a specified list of hosts or networks for onward transmission to any domain. Such hosts may also be identified by means of the SMTP AUTH mechanism. It is also possible to enable relaying to any domain whose MX records point to the local host, without having to list the domains explicitly.

### 20. Sender verification

The return path of a message (also known as the ‘envelope sender’) is used when Exim has to return an error message. If this is a bad address, the error message cannot be delivered, and the postmaster has to sort things out.

Sender verification (a configurable option that applies to SMTP input) is intended to pass this work to a foreign postmaster, by refusing to accept the message in the first place. There is an exception list which can specify certain hosts (with optional RFC 1413 identifications) that are allowed to bypass the check.

A certain amount of SPAM mail contains invalid return paths. Apart from this, there are two main causes: misconfigured mailers (gateways in particular), and users fooling around with mail. Sender verification catches both of them. It operates by passing the sender address through the directors and routers in verification mode; if this fails, the message is not accepted.

The first thing foreign postmasters ask when they learn about an apparently legitimate rejected message is 'What were the headers?'. For this reason, and also to collect evidence in cases of mail forgery, Exim does not initially reject a message after the MAIL FROM command in the SMTP session. It reads the message, so as to be able to write the headers to the rejection log, and then gives a hard error response to the sending host.

Unfortunately, several mailers believe that any error response after the data for a message has been sent indicates a temporary error. Consequently, such mailers will continue to try to send a message that has been rejected as described above. To prevent this, whenever a message is rejected, Exim records the time, bad address, and host in a DBM database. If the same host sends the same bad address within 24 hours, it is rejected immediately at the MAIL FROM command.

Sadly, even this doesn't stop some mailers from repeatedly trying to send the message. As a last resort, if the same host sends the same bad address for a third time in 24 hours, the MAIL FROM command is accepted, but all subsequent RCPT TO commands are rejected. If this does not stop a remote mailer then it is badly broken.

If the attempt to verify the sender address cannot be completed (typically because of a DNS timeout) Exim returns a temporary error code after the MAIL FROM command, which should cause the remote mailer to try again later. However, it is possible to configure Exim to accept the message in these circumstances.

Many messages with bad return paths in fact contain perfectly valid 'From' or 'Reply-to' headers. For administrators that want a quieter life, there is a configuration option which causes Exim to check these headers if the return path is bad, and if a good address is found, to use it to replace the return path. The old value is retained in an header whose name begins with 'X-'.

### 21. Sender lock out

More and more unsolicited junk mail is being seen on the Internet. It is sometimes useful to be able to reject messages (from any host) with particular sender addresses in the envelope. Exim can be configured to reject messages whose sender addresses match certain patterns, either by failing the MAIL FROM command, or (because some mailers take no notice of that) by failing all RCPT TO commands.

### 22. Receiver verification

Exim can be configured so that it checks the addresses given in incoming SMTP RCPT TO commands as they are received. A failing address can be immediately rejected, or it can be logged and accepted. If verification cannot be completed (typically because of a DNS timeout) either a temporary error code can be given, or the address can be logged and accepted.

### 23. The 'percent hack'

The so-called 'percent hack' is the feature of mailers whereby a local part containing a percent sign gets interpreted as an entire new address, with the percent replaced by @. This is used for explicit mail routing and sometimes for testing. In Exim, it is possible to configure which local domains, if any, allow the 'percent hack', though this is not recommended. Such usage, if configured, is, however, subject to the relay controls.

## 24. Security

Exim is written as a single binary that has to run setuid to root. I did start off trying to write it as a number of different modules, but soon came to the conclusion that, for this design of mailer, it was not worth it, because the functions don't decompose cleanly. For example, if you want to verify addresses while receiving mail you need all the directing and routing apparatus to be available.

Exim runs each local delivery in a separate process which is setuid to the relevant local user. In addition, it can be configured to run under a given non-root uid (and gid) for much of the rest of the time, and this is the recommended practice. In particular, it need not be root while sending or receiving SMTP mail.

Exim checks the permissions and owners of files to which messages are to be appended, and refuses to proceed with the delivery if things are not right.

Delivery of messages to pipes or files is supported only as a result of expanding an address via an alias or a forward file, provided this is permitted by the configuration. Externally generated local addresses cannot specify files or pipes – no special action is taken for addresses starting with the file or pipe characters, so they will usually fail.

Use of the VRFY, EXPN, and ETRN functions in SMTP connections is controlled by configuration options. The DEBUG function is not supported at all.

## 25. The Exim Monitor

A program for monitoring Exim and displaying information in an X window is provided. This can be configured to show stripcharts of incoming and outgoing mail in various categories. It also shows a 'tail' of the main log file, and information about messages on the queue.

There is a menu of operations that can be performed by suitably privileged users. Messages can be frozen, thawed, deleted, caused to be delivered, modified, or returned to their senders from this interface. However, all these actions can also be performed from the command line interface as well.

* * *