

Here are some useful things to know for managing an Exim 4 server. This assumes a prior working knowledge of SMTP, MTAs, and a UNIX shell prompt.

## Message-IDs and spool files

The message-IDs that Exim uses to refer to messages in its queue are mixed-case alphanumeric, and take the form of: XXXXXX-YYYYYY-ZZ. Most commands related to managing the queue and logging use these message-ids.

There are three -- count 'em, THREE -- files for each message in the spool directory. If you're dealing with these files by hand, instead of using the appropriate exim commands as detailed below, make sure you get them all, and don't leave Exim with remnants of messages in the queue. I used to mess directly with these files when I first started running Exim machines, but thanks to the utilities described below, I haven't needed to do that in many months.

Files in /var/spool/exim/msglog contain logging information for each message and are named the same as the message-id.

Files in /var/spool/exim/input are named after the message-id, plus a suffix denoting whether it is the envelope header (-H) or message data (-D).

These directories may contain further hashed subdirectories to deal with larger mail queues, so don't expect everything to always appear directly in the top /var/spool/exim/input or /var/spool/exim/msglog directories; any searches or greps will need to be recursive. See if there is a proper way to do what you're doing before working directly on the spool files.

## Basic information

Print a count of the messages in the queue:

```
root@localhost# exim -bpc
```

Print a listing of the messages in the queue (time queued, size, message-id, sender, recipient):

```
root@localhost# exim -bp
```

Print a summary of messages in the queue (count, volume, oldest, newest, domain, and totals):

```
root@localhost# exim -bp | exiqsumm
```

Print what Exim is doing right now:

```
root@localhost# exiwhat
```

Test how exim will route a given address:

```
root@localhost# exim -bt alias@localdomain.com
```

```
user@thishost.com
  <-- alias@localdomain.com
  router = localuser, transport = local_delivery
root@localhost# exim -bt user@thishost.com
user@thishost.com
  router = localuser, transport = local_delivery
root@localhost# exim -bt user@remotehost.com
  router = lookuphost, transport = remote_smtp
  host mail.remotehost.com [1.2.3.4] MX=0
```

Run a pretend SMTP transaction from the command line, as if it were coming from the given IP address. This will display Exim's checks, ACLs, and filters as they are applied. The message will NOT actually be delivered.

```
root@localhost# exim -bh 192.168.11.22
```

Display all of Exim's configuration settings:

```
root@localhost# exim -bP
```

## Searching the queue with exiqgrep

Exim includes a utility that is quite nice for grepping through the queue, called `exiqgrep`. [Learn it. Know it. Live it.](#) If you're not using this, and if you're not familiar with the various flags it uses, you're probably doing things the hard way, like piping ``exim -bp`` into `awk`, `grep`, `cut`, or ``wc -l``. Don't make life harder than it already is.

First, various flags that control what messages are matched. These can be combined to come up with a very particular search.

Use `-f` to search the queue for messages from a specific sender:

```
root@localhost# exiqgrep -f [luser]@domain
```

Use `-r` to search the queue for messages for a specific recipient/domain:

```
root@localhost# exiqgrep -r [luser]@domain
```

Use `-o` to print messages older than the specified number of seconds. For example, messages older than 1 day:

```
root@localhost# exiqgrep -o 86400 [...]
```

Use `-y` to print messages that are younger than the specified number of seconds. For example, messages less than an hour old:

```
root@localhost# exiqgrep -y 3600 [...]
```

Use `-s` to match the size of a message with a regex. For example, 700-799 bytes:

```
root@localhost# exiqgrep -s '^7..$' [...]
```

Use `-z` to match only frozen messages, or `-x` to match only unfrozen messages.

There are also a few flags that control the display of the output.

Use `-i` to print just the message-id as a result of one of the above two searches:

```
root@localhost# exiqgrep -i [ -r | -f ] ...
```

Use `-c` to print a count of messages matching one of the above searches:

```
root@localhost# exiqgrep -c ...
```

Print just the message-id of the entire queue:

```
root@localhost# exiqgrep -i
```

## Managing the queue

The main exim binary (`/usr/sbin/exim`) is used with various flags to make things happen to messages in the queue. Most of these require one or more message-IDs to be specified in the command line, which is where `exiqgrep -i`` as described above really comes in handy.

Start a queue run:

```
root@localhost# exim -q -v
```

Start a queue run for just local deliveries:

```
root@localhost# exim -ql -v
```

Remove a message from the queue:

```
root@localhost# exim -Mrm <message-id> [ <message-id> ... ]
```

Freeze a message:

```
root@localhost# exim -Mf <message-id> [ <message-id> ... ]
```

Thaw a message:

```
root@localhost# exim -Mt <message-id> [ <message-id> ... ]
```

Deliver a message:

```
root@localhost# exim -M <message-id> [ <message-id> ... ]
```

Force a message to fail and bounce as "cancelled by administrator":

```
root@localhost# exim -Mg <message-id> [ <message-id> ... ]
```

Remove all frozen messages:

```
root@localhost# exiqgrep -z -i | xargs exim -Mrm
```

Remove all messages older than five days ( $86400 * 5 = 432000$  seconds):

```
root@localhost# exiqgrep -o 432000 -i | xargs exim -Mrm
```

Freeze all queued mail from a given sender:

```
root@localhost# exiqgrep -i -f luser@example.tld | xargs exim -Mf
```

View a message's headers:

```
root@localhost# exim -Mvh <message-id>
```

View a message's body:

```
root@localhost# exim -Mvb <message-id>
```

View a message's logs:

```
root@localhost# exim -Mvl <message-id>
```

Add a recipient to a message:

```
root@localhost# exim -Mar <message-id> <address> [ <address> ... ]
```

Edit the sender of a message:

```
root@localhost# exim -Mes <message-id> <address>
```

## Access control

Exim allows you to apply [access control lists](#) at various points of the SMTP transaction by specifying an ACL to use and defining its conditions in `exim.conf`. You could start with the HELO string.

```
# Specify the ACL to use after HELO
acl_smtp_helo = check_helo

# Conditions for the check_helo ACL:
check_helo:

    deny message = Gave HELO/EHLO as "friend"
    log_message = HELO/EHLO friend
    condition = ${if eq {$sender_helo_name}{friend} {yes}{no}}

    deny message = Gave HELO/EHLO as our IP address
    log_message = HELO/EHLO our IP address
    condition = ${if eq {$sender_helo_name}{$interface_address}
{yes}{no}}

    accept
```

NOTE: Pursue HELO checking at your own peril. The HELO is fairly unimportant in the grand scheme of SMTP these days, so don't put too much faith in whatever it contains. Some spam might seem to use a telltale HELO string, but you might be

surprised at how many legitimate messages start off with a questionable HELO as well. Anyway, it's just as easy for a spammer to send a proper HELO than it is to send HELO im.a.spammer, so consider yourself lucky if you're able to stop much spam this way.

Next, you can perform a check on the sender address or remote host. This shows how to do that after the RCPT TO command; if you reject here, as opposed to rejecting after the MAIL FROM, you'll have better data to log, such as who the message was intended for.

```
# Specify the ACL to use after RCPT TO
acl_smtp_rcpt = check_recipient

# Conditions for the check_recipient ACL
check_recipient:

    # [...]

    drop hosts = /etc/exim_reject_hosts
    drop senders = /etc/exim_reject_senders

    # [ Probably a whole lot more... ]
```

This example uses two plain text files as blacklists. Add appropriate entries to these files - hostnames/IP addresses to /etc/exim\_reject\_hosts, addresses to /etc/exim\_reject\_senders, one entry per line.

It is also possible to perform [content scanning](#) using a regex against the body of a message, though obviously this can cause Exim to use more CPU than it otherwise would need to, especially on large messages.

```
# Specify the ACL to use after DATA
acl_smtp_data = check_message

# Conditions for the check_messages ACL
check_message:

    deny message = "Sorry, Charlie: $regex_match_string"
    regex = ^Subject:: .*Lower your self-esteem by becoming a sysadmin

    accept
```

## Fix SMTP-Auth for Pine

If pine can't use SMTP authentication on an Exim host and just returns an "unable to authenticate" message without even asking for a password, add the following line to exim.conf:

```
begin authenticators

    fixed_plain:
    driver = plaintext
    public_name = PLAIN
    server_condition = "${perl{checkuserpass}}{$1}{$2}{$3}"
    server_set_id = $2
> server_prompts = :
```

This was a problem on CPanel Exim builds awhile ago, but they seem to have added this line to their current stock configuration.

## Log the subject line

This is one of the most useful configuration tweaks I've ever found for Exim. Add this to `exim.conf`, and you can log the subject lines of messages that pass through your server. This is great for troubleshooting, and for getting a very rough idea of what messages may be spam.

```
log_selector = +subject
```

[Reducing or increasing what is logged.](#)

## Disable identd lookups

Frankly, I don't think [identd](#) has been useful for a long time, if ever. Identd relies on the connecting host to confirm the identity (system UID) of the remote user who owns the process that is making the network connection. This may be of some use in the world of shell accounts and IRC users, but it really has no place on a high-volume SMTP server, where the UID is often simply "mail" or whatever the remote MTA runs as, which is useless to know. It's overhead, and results in nothing but delays while the identd query is refused or times out. You can stop your Exim server from making these queries by setting the timeout to zero seconds in `exim.conf`:

```
rfc1413_query_timeout = 0s
```

## Disable Attachment Blocking

To disable the executable-attachment blocking that many Cpanel servers do by default but don't provide any controls for on a per-domain basis, add the following block to the beginning of the `/etc/antivirus.exim` file:

```
if $header_to: matches "example\.com|example2\.com"
then
    finish
endif
```

It is probably possible to use a separate file to list these domains, but I haven't had to do this enough times to warrant setting such a thing up.

## Searching the logs with exigrep

The [exigrep](#) utility (not to be confused with `exiqgrep`) is used to search an exim log for a string or pattern. It will print all log entries with the same internal message-id as those that matched the pattern, which is very handy since any message will take up at least three lines in the log. `exigrep` will search the entire content of a log entry, not just particular fields.

One can search for messages sent from a particular IP address:

```
root@localhost# exigrep '<= .* \[12.34.56.78\]' /path/to/exim_log
```

Search for messages sent to a particular IP address:

```
root@localhost# exigrep '=> .* \[12.34.56.78\]' /path/to/exim_log
```

This example searches for outgoing messages, which have the "=>" symbol, sent to "user@domain.tld". The pipe to grep for the "<=" symbol will match only the lines with information on the sender - the From address, the sender's IP address, the message size, the message ID, and the subject line if you have enabled logging the subject. The purpose of doing such a search is that the desired information is not on the same log line as the string being searched for.

```
root@localhost# exigrep '=> .*user@domain.tld' /path/to/exim_log |  
fgrep '<='
```

Generate and display Exim stats from a logfile:

```
root@localhost# eximstats /path/to/exim_mainlog
```

Same as above, with less verbose output:

```
root@localhost# eximstats -ne -nr -nt /path/to/exim_mainlog
```

Same as above, for one particular day:

```
root@localhost# fgrep YYYY-MM-DD /path/to/exim_mainlog | eximstats
```

## Bonus!

To delete all queued messages containing a certain string in the body:

```
root@localhost# grep -lr 'a certain string' /var/spool/exim/input/ | \  
sed -e 's/^\.*\([a-zA-Z0-9-]*\)-[DH]$/\1/g' | xargs  
exim -Mrm
```

Note that the above only delves into /var/spool/exim in order to grep for queue files with the given string, and that's just because exiqgrep doesn't have a feature to grep the actual bodies of messages. If you are deleting these files directly, **YOU ARE DOING IT WRONG!** Use the appropriate exim command to properly deal with the queue.

If you have to feed many, many message-ids (such as the output of an `exiqgrep -i` command that returns a lot of matches) to an exim command, you may exhaust the limit of your shell's command line arguments. In that case, pipe the listing of message-ids into xargs to run only a limited number of them at once. For example, to remove thousands of messages sent from joe@example.com:

```
root@localhost# exiqgrep -i -f '<joe@example.com>' | xargs exim -Mrm
```

**Speaking of "DOING IT WRONG" -- Attention, CPanel forum readers**

I get a number of hits to this page from a link in [this post](#) at the CPanel forums. The question is:

Due to spamming, spoofing from fields, etc., etc., etc., I am finding it necessary to spend more time to clear the exim queue from time to time. [...] what command would I use to delete the queue

The answer is: Just turn exim off, because your customers are better off knowing that email simply isn't running on your server, than having their queued messages deleted without notice.

Or, figure out what is happening. The examples given in that post pay no regard to the legitimacy of any message, they simply delete everything, making the presumption that if a message is in the queue, it's junk. That is total fallacy. There are a number of reasons legitimate mail can end up in the queue. Maybe your backups or CPanel's "upcp" process are running, and your load average is high -- exim goes into a queue-only mode at a certain threshold, where it stops trying to deliver messages as they come in and just queues them until the load goes back down. Or, maybe it's an outgoing message, and the DNS lookup failed, or the connection to the domain's MX failed, or maybe the remote MX is busy or greylisting you with a 4xx deferral. These are all temporary failures, not permanent ones, and the whole point of having temporary failures in SMTP and a mail queue in your MTA is to be able to try again after awhile.

Exim already purges messages from the queue after the period of time specified in exim.conf. If you have this value set appropriately, there is absolutely no point in removing everything from your queue every day with a cron job. **You will lose legitimate mail, and the sender and recipient will never know if or why it happened.** Do not do this!

If you regularly have a large number of messages in your queue, find out why they are there. If they are outbound messages, see who is sending them, where they're addressed to, and why they aren't getting there. If they are inbound messages, find out why they aren't getting delivered to your user's account. If you need to delete some, use `exiqgrep` to pick out just the ones that should be deleted.

## Reload the configuration

After making changes to exim.conf, you need to give the main exim pid a SIGHUP to re-exec it and have the configuration re-read. Sure, you could stop and start the service, but that's overkill and causes a few seconds of unnecessary downtime. Just do this:

```
root@localhost# kill -HUP `cat /var/spool/exim/exim-daemon.pid`
```

You should then see something resembling the following in `exim_mainlog`:

```
pid 1079: SIGHUP received: re-exec daemon
exim 4.52 daemon started: pid=1079, -qlh, listening for SMTP on port
25 (IPv4)
```

## Read The Fucking Manual

[The Exim Home Page](#)

[Documentation For Exim](#)

[The Exim Specification - Version 4.5x](#)

[Exim command line arguments](#)