

Section 42 - System-wide message filtering

42. System-wide message filtering

The previous chapters (on ACLs and the local scan function) describe checks that can be applied to messages before they are accepted by a host. There is also a mechanism for checking messages once they have been received, but before they are delivered. This is called the system filter.

The system filter operates in a similar manner to users' filter files, but it is run just once per message (however many recipients the message has). It should not normally be used as a substitute for routing, because deliver commands in a system router provide new envelope recipient addresses. The system filter must be an Exim filter. It cannot be a Sieve filter.

The system filter is run at the start of a delivery attempt, before any routing is done. If a message fails to be completely delivered at the first attempt, the system filter is run again at the start of every retry. If you want your filter to do something only once per message, you can make use of the `first_delivery` condition in an `if` command in the filter to prevent it happening on retries.

Warning: Because the system filter runs just once, variables that are specific to individual recipient addresses, such as `$local_part` and `$domain`, are not set, and the `“personal”` condition is not meaningful. If you want to run a centrally-specified filter for each recipient address independently, you can do so by setting up a suitable redirect router, as described in section 42.8 below.

42.1 Specifying a system filter

The name of the file that contains the system filter must be specified by setting `system_filter`. If you want the filter to run under a uid and gid other than root, you must also set `system_filter_user` and `system_filter_group` as appropriate. For example:

```
system_filter = /etc/mail/exim.filter
system_filter_user = exim
```

If a system filter generates any deliveries directly to files or pipes (via the `save` or `pipe` commands), transports to handle these deliveries must be specified by setting `system_filter_file_transport` and `system_filter_pipe_transport`, respectively. Similarly, `system_filter_reply_transport` must be set to handle any messages generated by the `reply` command.

42.2 Testing a system filter

You can run simple tests of a system filter in the same way as for a user filter, but you should use `-bF` rather than `-bf`, so that features that are permitted only in system filters are recognized.

If you want to test the combined effect of a system filter and a user filter, you can use both `-bF` and `-bf` on the same command line.

42.3 Contents of a system filter

The language used to specify system filters is the same as for users' filter files. It is described in the separate end-user document Exim's interface to mail filtering. However, there are some additional features that are available only in system filters; these are described in subsequent sections. If they are encountered in a user's filter file or when testing with `-bf`, they cause errors.

There are two special conditions which, though available in users' filter files, are designed for use in system filters. The condition `first_delivery` is true only for the first attempt at delivering a message, and `manually_thawed` is true only if the message has been frozen, and subsequently thawed by an admin user. An explicit forced delivery counts as a manual thaw, but thawing as a result of the `auto_thaw` setting does not.

Warning: If a system filter uses the `first_delivery` condition to specify an `“unseen”` (non-significant) delivery, and that delivery does not succeed, it will not be tried again. If you want Exim to retry an unseen delivery until it succeeds, you should arrange to set it up every time the filter runs.

When a system filter finishes running, the values of the variables `$n0` – `$n9` are copied into `$sn0` – `$sn9` and are thereby made available to users' filter files. Thus a system filter can, for example, set up `“scores”`; to which users' filter files can refer.

42.4 Additional variable for system filters

The expansion variable `$recipients`, containing a list of all the recipients of the message (separated by commas and white space), is available in system filters. It is not available in users' filters for privacy reasons.

42.5 Defer, freeze, and fail commands for system filters

There are three extra commands (`defer`, `freeze` and `fail`) which are always available in system filters, but are not normally enabled in users' filters. (See the `allow_defer`, `allow_freeze` and `allow_fail` options for the redirect router.)

These commands can optionally be followed by the word `text` and a string containing an error message, for example: `fail text "this message looks like spam to me"`

The keyword `text` is optional if the next character is a double quote.

The `defer` command defers delivery of the original recipients of the message. The `fail` command causes all the original recipients to be failed, and a bounce message to be created. The `freeze` command suspends all delivery attempts for the original recipients. In all cases, any new deliveries that are specified by the filter are attempted as normal after the filter has run.

The `freeze` command is ignored if the message has been manually unfrozen and not manually frozen since. This means that automatic freezing by a system filter can be used as a way of checking out suspicious messages. If a message is found to be all right, manually unfreezing it allows it to be delivered.

The text given with a `fail` command is used as part of the bounce message as well as being written to the log. If the message is quite long, this can fill up a lot of log space when such failures are common. To reduce the size of the log message, Exim interprets the text in a special way if it starts with the two characters `<<` and contains `>>` later. The text between these two strings is written to the log, and the rest of the text is used in the bounce message. For example: `fail "<<filter test 1>>Your message is rejected \ because it contains attachments that we are \ not prepared to receive."`

Take great care with the `fail` command when basing the decision to fail on the contents of the message, because the bounce message will of course include the contents of the original message and will therefore trigger the `fail` command again (causing a mail loop) unless steps are taken to prevent this. Testing the `error_message` condition is one way to prevent this. You could use, for example `if $message_body contains "this is spam" and not error_message then fail text "spam is not wanted here" endif`

though of course that might let through unwanted bounce messages. The alternative is clever checking of the body and/or headers to detect bounces generated by the filter.

The interpretation of a system filter file ceases after a `defer`, `freeze`, or `fail` command is obeyed. However, any deliveries that were set up earlier in the filter file are honoured, so you can use a sequence such as `mail ... freeze`

to send a specified message when the system filter is freezing (or deferring or failing) a message. The normal deliveries for the message do not, of course, take place. 42.6 Adding and removing headers in a system filter

Two filter commands that are available only in system filters are: `headers add <string>`
`headers remove <string>`

The argument for the `headers add` is a string that is expanded and then added to the end of the message's headers. It is the responsibility of the filter maintainer to make sure it conforms to RFC 2822 syntax. Leading white space is ignored, and if the string is otherwise empty, or if the expansion is forced to fail, the command has no effect.

You can use `“\n”` within the string, followed by white space, to specify continued header lines. More than one header may be added in one command by including `“\n”` within the string without any following white space. For example: `headers add "X-header-1:\n \ continuation of X-header-1 ...\n \ X-header-2:"`

Note that the header line continuation white space after the first newline must be placed before the backslash that continues the input string, because white space after input continuations is ignored.

The argument for `headers remove` is a colon-separated list of header names. This command applies only to those headers that are stored with the message; those that are added at delivery time (such as `Envelope-To:` and `Return-Path:`) cannot be removed by this means. If there is more than one header with the same name, they are all removed.

The `headers` command in a system filter makes an immediate change to the set of header lines that was received with

the message (with possible additions from ACL processing). Subsequent commands in the system filter operate on the modified set, which also forms the basis for subsequent message delivery. Unless further modified during routing or transporting, this set of headers is used for all recipients of the message.

During routing and transporting, the variables that refer to the contents of header lines refer only to those lines that are in this set. Thus, header lines that are added by a system filter are visible to users' filter files and to all routers and transports. This contrasts with the manipulation of header lines by routers and transports, which is not immediate, but which instead is saved up until the message is actually being written (see section 43.17).

If the message is not delivered at the first attempt, header lines that were added by the system filter are stored with the message, and so are still present at the next delivery attempt. Header lines that were removed are still present, but marked “deleted” so that they are not transported with the message. For this reason, it is usual to make the headers command conditional on `first_delivery` so that the set of header lines is not modified more than once.

Because header modification in a system filter acts immediately, you have to use an indirect approach if you want to modify the contents of a header line. For example: headers add "Old-Subject: \$h_subject:"

```
headers remove "Subject"
```

```
headers add "Subject: new subject (was: $h_old-subject:)"
```

```
headers remove "Old-Subject"
```

42.7 Setting an errors address in a system filter

In a system filter, if a `deliver` command is followed by `errors_to <some address>`

in order to change the envelope sender (and hence the error reporting) for that delivery, any address may be specified. (In a user filter, only the current user's address can be set.) For example, if some mail is being monitored, you might use `unseen deliver monitor@spying.example errors_to root@local.example`

to take a copy which would not be sent back to the normal error reporting address if its delivery failed. 42.8 Per-address filtering

In contrast to the system filter, which is run just once per message for each delivery attempt, it is also possible to set up a system-wide filtering operation that runs once for each recipient address. In this case, variables such as `$local_part` and `$domain` can be used, and indeed, the choice of filter file could be made dependent on them. This is an example of a router which implements such a filter: `central_filter`:

```
check_local_user
```

```
driver = redirect
```

```
domains = +local_domains
```

```
file = /central/filters/$local_part
```

```
no_verify
```

```
allow_filter
```

```
allow_freeze
```

The filter is run in a separate process under its own uid. Therefore, either `check_local_user` must be set (as above), in which case the filter is run as the local user, or the `user` option must be used to specify which user to use. If both are set, `user` overrides. Care should be taken to ensure that none of the commands in the filter file specify a significant delivery if the message is to go on to be delivered to its intended recipient. The router will not then claim to have dealt with the address, so it will be passed on to subsequent routers to be delivered in the normal way.