

## Section 24 - Generic options for transports

### 24. Generic options for transports

The following generic options apply to all transports:

`body_onlyUse: transportsType: booleanDefault: false`

If this option is set, the message's headers are not transported. It is mutually exclusive with `headers_only`. If it is used with the `appendfile` or `pipe` transports, the settings of `message_prefix` and `message_suffix` should be checked, because this option does not automatically suppress them.

`current_directoryUse: transportsType: stringDefault: unset`

This specifies the current directory that is to be set while running the transport, overriding any value that may have been set by the router. If the expansion fails for any reason, including forced failure, an error is logged, and delivery is deferred.

`disable_loggingUse: transportsType: booleanDefault: false`

If this option is set true, nothing is logged for any deliveries by the transport or for any transport errors. You should not set this option unless you really, really know what you are doing.

`debug_printUse: transportsType: stringDefault: unset`

If this option is set and debugging is enabled (see the `-d` command line option), the string is expanded and included in the debugging output when the transport is run. If expansion of the string fails, the error message is written to the debugging output, and Exim carries on processing. This facility is provided to help with checking out the values of variables and so on when debugging driver configurations. For example, if a `headers_add` option is not working properly, `debug_print` could be used to output the variables it references. A newline is added to the text if it does not end with one.

`delivery_date_addUse: transportsType: booleanDefault: false`

If this option is true, a `Delivery-date:` header is added to the message. This gives the actual time the delivery was made. As this is not a standard header, Exim has a configuration option (`delivery_date_remove`) which requests its removal from incoming messages, so that delivered messages can safely be resent to other recipients.

`driverUse: transportsType: stringDefault: unset`

This specifies which of the available transport drivers is to be used. There is no default, and this option must be set for every transport.

`envelope_to_addUse: transportsType: booleanDefault: false`

If this option is true, an `Envelope-to:` header is added to the message. This gives the original address(es) in the incoming envelope that caused this delivery to happen. More than one address may be present if the transport is configured to handle several addresses at once, or if more than one original address was redirected to the same final address. As this is not a standard header, Exim has a configuration option (`envelope_to_remove`) which requests its removal from incoming messages, so that delivered messages can safely be resent to other recipients.

`groupUse: transportsType: stringDefault: Exim group`

This option specifies a gid for running the transport process, overriding any value that the router supplies, and also overriding any value associated with user (see below).

`headers_addUse: transportsType: stringDefault: unset`

This option specifies a string of text that is expanded and added to the header portion of a message as it is transported, as described in section 43.17. Additional header lines can also be specified by routers. If the result of the expansion is an empty string, or if the expansion is forced to fail, no action is taken. Other expansion failures are treated as errors and cause the delivery to be deferred.

`headers_onlyUse: transportsType: booleanDefault: false`

If this option is set, the message's body is not transported. It is mutually exclusive with `body_only`. If it is used with the `appendfile` or `pipe` transports, the settings of `message_prefix` and `message_suffix` should be checked, since this option does not automatically suppress them.

`headers_removeUse: transportsType: string&dagger;Default: unset`

This option specifies a string that is expanded into a list of header names; these headers are omitted from the message as it is transported, as described in section 43.17. Header removal can also be specified by routers. If the result of the expansion is an empty string, or if the expansion is forced to fail, no action is taken. Other expansion failures are treated as errors and cause the delivery to be deferred.

`headers_rewriteUse: transportsType: stringDefault: unset`

This option allows addresses in header lines to be rewritten at transport time, that is, as the message is being copied to its destination. The contents of the option are a colon-separated list of rewriting rules. Each rule is in exactly the same form as one of the general rewriting rules that are applied when a message is received. These are described in chapter 31. For example, `headers_rewrite = a@b c@d f : \`

`x@y w@z`

changes `a@b` into `c@d` in `From:` header lines, and `x@y` into `w@z` in all address-bearing header lines. The rules are applied to the header lines just before they are written out at transport time, so they affect only those copies of the message that pass through the transport. However, only the message's original header lines, and any that were added by a system filter, are rewritten. If a router or transport adds header lines, they are not affected by this option. These rewriting rules are not applied to the envelope. You can change the return path using `return_path`, but you cannot change envelope recipients at this time.

`home_directoryUse: transportsType: string&dagger;Default: unset`

This option specifies a home directory setting for a local transport, overriding any value that may be set by the router. The home directory is placed in `$home` while expanding the transport's private options. It is also used as the current directory if no current directory is set by the `current_directory` option on the transport or the `transport_current_directory` option on the router. If the expansion fails for any reason, including forced failure, an error is logged, and delivery is deferred.

`initgroupsUse: transportsType: booleanDefault: false`

If this option is true and the uid for the delivery process is provided by the transport, the `initgroups()` function is called when running the transport to ensure that any additional groups associated with the uid are set up.

`message_size_limitUse: transportsType: string&dagger;Default: 0`

This option controls the size of messages passed through the transport. It is expanded before use; the result of the expansion must be a sequence of digits, optionally followed by K or M. If the expansion fails for any reason, including forced failure, or if the result is not of the required form, delivery is deferred. If the value is greater than zero and the size of a message exceeds this limit, the address is failed. If there is any chance that the resulting bounce message could be routed to the same transport, you should ensure that `return_size_limit` is less than the transport's `message_size_limit`, as otherwise the bounce message will fail to get delivered.

`rcpt_include_affixesUse: transportsType: booleanDefault: false`

When this option is false (the default), and an address that has had any affixes (prefixes or suffixes) removed from the local part is delivered by any form of SMTP or LMTP, the affixes are not included. For example, if a router that contains `local_part_prefix = *-`

routes the address `abc-xyz@some.domain` to an SMTP transport, the envelope is delivered with `RCPT TO:<xyz@some.domain>`

This is also the case when an ACL-time callout is being used to verify a recipient address. However, if `rcpt_include_affixes` is set true, the whole local part is included in the RCPT command. This option applies to BSMTP deliveries by the `appendfile` and `pipe` transports as well as to the `lmtp` and `smtp` transports.

`retry_use_local_partUse: transportsType: booleanDefault: see below`

When a delivery suffers a temporary failure, a retry record is created in Exim's hints database. For remote deliveries, the key for the retry record is based on the name and/or IP address of the failing remote host. For local deliveries, the key is normally the entire address, including both the local part and the domain. This is suitable for most common cases of local delivery temporary failure &ndash; for example, exceeding a mailbox quota should delay only deliveries to that mailbox, not to the whole domain.

However, in some special cases you may want to treat a temporary local delivery as a failure associated with the domain, and not with a particular local part. (For example, if you are storing all mail for some domain in files.) You can do this by setting `retry_use_local_part` false.

For all the local transports, its default value is true. For remote transports, the default value is false for tidiness, but changing the value has no effect on a remote transport in the current implementation.

`return_path`Use: transportsType: string&dagger;Default: unset

If this option is set, the string is expanded at transport time and replaces the existing return path (envelope sender) value in the copy of the message that is being delivered. An empty return path is permitted. This feature is designed for remote deliveries, where the value of this option is used in the SMTP MAIL command. If you set `return_path` for a local transport, the only effect is to change the address that is placed in the Return-path: header line, if one is added to the message (see the next option).

The expansion can refer to the existing value via `$return_path`. This is either the message's envelope sender, or an address set by the `errors_to` option on a router. If the expansion is forced to fail, no replacement occurs; if it fails for another reason, delivery is deferred. This option can be used to support VERP (Variable Envelope Return Paths) &ndash; see section 46.6.

Note: If a delivery error is detected locally, including the case when a remote server rejects a message at SMTP time, the bounce message is not sent to the value of this option. It is sent to the previously set errors address. This defaults to the incoming sender address, but can be changed by setting `errors_to` in a router.

`return_path_add`Use: transportsType: booleanDefault: false

If this option is true, a Return-path: header is added to the message. Although the return path is normally available in the prefix line of BSD mailboxes, this is commonly not displayed by MUAs, and so the user does not have easy access to it.

RFC 2821 states that the Return-path: header is added to a message &ldquo;when the delivery SMTP server makes the final delivery&rdquo;. This implies that this header should not be present in incoming messages. Exim has a configuration option, `return_path_remove`, which requests removal of this header from incoming messages, so that delivered messages can safely be resent to other recipients.

`shadow_condition`Use: transportsType: string&dagger;Default: unset

See `shadow_transport` below.

`shadow_transport`Use: transportsType: stringDefault: unset

A local transport may set the `shadow_transport` option to the name of another local transport. Shadow remote transports are not supported.

Whenever a delivery to the main transport succeeds, and either `shadow_condition` is unset, or its expansion does not result in the empty string or one of the strings &ldquo;0&rdquo; or &ldquo;no&rdquo; or &ldquo>false&rdquo;, the message is also passed to the shadow transport, with the same delivery address or addresses. If expansion fails, no action is taken except that non-forced expansion failures cause a log line to be written.

The result of the shadow transport is discarded and does not affect the subsequent processing of the message. Only a single level of shadowing is provided; the `shadow_transport` option is ignored on any transport when it is running as a shadow. Options concerned with output from pipes are also ignored. The log line for the successful delivery has an item added on the end, of the form `ST=<shadow transport name>`

If the shadow transport did not succeed, the error message is put in parentheses afterwards. Shadow transports can be used for a number of different purposes, including keeping more detailed log information than Exim normally provides, and implementing automatic acknowledgement policies based on message headers that some sites insist on.

transport\_filterUse: transportsType: string&dagger;Default: unset

This option sets up a filtering (in the Unix shell sense) process for messages at transport time. It should not be confused with mail filtering as set up by individual users or via a system filter.

When the message is about to be written out, the command specified by transport\_filter is started up in a separate, parallel process, and the entire message, including the header lines, is passed to it on its standard input (this in fact is done from a third process, to avoid deadlock). The command must be specified as an absolute path.

The lines of the message that are written to the transport filter are terminated by newline (&ldquo;\n&rdquo;). The message is passed to the filter before any SMTP-specific processing, such as turning &ldquo;\n&rdquo; into &ldquo;\r\n&rdquo; and escaping lines beginning with a dot, and also before any processing implied by the settings of check\_string and escape\_string in the appendfile or pipe transports.

The standard error for the filter process is set to the same destination as its standard output; this is read and written to the message&rsquo;s ultimate destination. The process that writes the message to the filter, the filter itself, and the original process that reads the result and delivers it are all run in parallel, like a shell pipeline.

The filter can perform any transformations it likes, but of course should take care not to break RFC 2822 syntax. A demonstration Perl script is provided in util/transport-filter.pl; this makes a few arbitrary modifications just to show the possibilities. Exim does not check the result, except to test for a final newline when SMTP is in use. All messages transmitted over SMTP must end with a newline, so Exim supplies one if it is missing.

A transport filter can be used to provide content-scanning on a per-user basis at delivery time if the only required effect of the scan is to modify the message. For example, a content scan could insert a new header line containing a spam score. This could be interpreted by a filter in the user&rsquo;s MUA. It is not possible to discard a message at this stage.

A problem might arise if the filter increases the size of a message that is being sent down an SMTP connection. If the receiving SMTP server has indicated support for the SIZE parameter, Exim will have sent the size of the message at the start of the SMTP session. If what is actually sent is substantially more, the server might reject the message. This can be worked round by setting the size\_addition option on the smtp transport, either to allow for additions to the message, or to disable the use of SIZE altogether.

The value of the transport\_filter option is the command string for starting the filter, which is run directly from Exim, not under a shell. The string is parsed by Exim in the same way as a command string for the pipe transport: Exim breaks it up into arguments and then expands each argument separately (see section 29.3). Any kind of expansion failure causes delivery to be deferred. The special argument \$pipe\_addresses is replaced by a number of arguments, one for each address that applies to this delivery. (This isn&rsquo;t an ideal name for this feature here, but as it was already implemented for the pipe transport, it seemed sensible not to change it.)

The expansion variables \$host and \$host\_address are available when the transport is a remote one. They contain the name and IP address of the host to which the message is being sent. For example: transport\_filter = /some/directory/transport-filter.pl \  
\$host \$host\_address \$sender\_address \$pipe\_addresses

Two problems arise if you want to use more complicated expansion items to generate transport filter commands, both of which due to the fact that the command is split up before expansion.

If an expansion item contains white space, you must quote it, so that it is all part of the same command item. If the entire option setting is one such expansion item, you have to take care what kind of quoting you use. For example: transport\_filter = '/bin/cmd\${if eq{\$host}{a.b.c}{1}{2}}'

This runs the command /bin/cmd1 if the host name is a.b.c, and /bin/cmd2 otherwise. If double quotes had been used, they would have been stripped by Exim when it read the option&rsquo;s value. When the value is used, if the single quotes were missing, the line would be split into two items, /bin/cmd\${if and eq{\$host}{a.b.c}{1}{2}}, and an error would occur when Exim tried to expand the first one.

Except for the special case of \$pipe\_addresses that is mentioned above, an expansion cannot generate multiple arguments, or a command name followed by arguments. Consider this example: transport\_filter = \${lookup{\$host}lsearch{/some/file}}

```
{${value}}{/bin/cat}}
```

The result of the lookup is interpreted as the name of the command, even if it contains white space. The simplest way round this is to use a shell: `transport_filter = /bin/sh -c ${lookup{$host}lsearch{/some/file}\  
{${value}}{/bin/cat}}`

The filter process is run under the same uid and gid as the normal delivery. For remote deliveries this is the Exim uid/gid by default. The command should normally yield a zero return code. Transport filters are not supposed to fail. A non-zero code is taken to mean that the transport filter encountered some serious problem. Delivery of the message is deferred; the message remains on the queue and is tried again later. It is not possible to cause a message to be bounced from a transport filter.

If a transport filter is set on an autoreply transport, the original message is passed through the filter as it is being copied into the newly generated message, which happens if the `return_message` option is set.

`transport_filter_timeout`Use: transportsType: timeDefault: 5m

When Exim is reading the output of a transport filter, it applies a timeout that can be set by this option. Exceeding the timeout is normally treated as a temporary delivery failure. However, if a transport filter is used with a pipe transport, a timeout in the transport filter is treated in the same way as a timeout in the pipe command itself. By default, a timeout is a hard error, but if the pipe transport's `timeout_defer` option is set true, it becomes a temporary error.

`user`Use: transportsType: string&dagger;Default: Exim user

This option specifies the user under whose uid the delivery process is to be run, overriding any uid that may have been set by the router. If the user is given as a name, the uid is looked up from the password data, and the associated group is taken as the value of the gid to be used if the `group` option is not set.

For deliveries that use local transports, a user and group are normally specified explicitly or implicitly (for example, as a result of `check_local_user`) by the router or transport. For remote transports, you should leave this option unset unless you really are sure you know what you are doing. When a remote transport is running, it needs to be able to access Exim's hints databases, because each host may have its own retry data.