

## Section 31 - Address rewriting

### 31. Address rewriting

There are some circumstances in which Exim automatically rewrites domains in addresses. The two most common are when an address is given without a domain (referred to as an &ldquo;unqualified address&rdquo;) or when an address contains an abbreviated domain that is expanded by DNS lookup.

Unqualified envelope addresses are accepted only for locally submitted messages, or for messages that are received from hosts matching `sender_unqualified_hosts` or `recipient_unqualified_hosts`, as appropriate. Unqualified addresses in header lines are qualified if they are in locally submitted messages, or messages from hosts that are permitted to send unqualified envelope addresses. Otherwise, unqualified addresses in header lines are neither qualified nor rewritten.

One situation in which Exim does not automatically rewrite a domain is when it is the name of a CNAME record in the DNS. The older RFCs suggest that such a domain should be rewritten using the &ldquo;canonical&rdquo; name, and some MTAs do this. The new RFCs do not contain this suggestion.

#### 31.1 Explicitly configured address rewriting

This chapter describes the rewriting rules that can be used in the main rewrite section of the configuration file, and also in the generic `headers_rewrite` option that can be set on any transport.

Some people believe that configured address rewriting is a Mortal Sin. Others believe that life is not possible without it. Exim provides the facility; you do not have to use it.

The main rewriting rules that appear in the &ldquo;rewrite&rdquo; section of the configuration file are applied to addresses in incoming messages, both envelope addresses and addresses in header lines. Each rule specifies the types of address to which it applies.

Rewriting of addresses in header lines applies only to those headers that were received with the message, and, in the case of transport rewriting, those that were added by a system filter. That is, it applies only to those headers that are common to all copies of the message. Header lines that are added by individual routers or transports (and which are therefore specific to individual recipient addresses) are not rewritten.

In general, rewriting addresses from your own system or domain has some legitimacy. Rewriting other addresses should be done only with great care and in special circumstances. The author of Exim believes that rewriting should be used sparingly, and mainly for &ldquo;regularizing&rdquo; addresses in your own domains. Although it can sometimes be used as a routing tool, this is very strongly discouraged.

There are two commonly encountered circumstances where rewriting is used, as illustrated by these examples:

-

The company whose domain is `hitch.fict.example` has a number of hosts that exchange mail with each other behind a firewall, but there is only a single gateway to the outer world. The gateway rewrites `*.hitch.fict.example` as `hitch.fict.example` when sending mail off-site.

-

A host rewrites the local parts of its own users so that, for example, `fp42@hitch.fict.example` becomes `Ford.Prefect@hitch.fict.example`.

#### 31.2 When does rewriting happen?

Configured address rewriting can take place at several different stages of a message's processing.

At the start of an ACL for MAIL, the sender address may have been rewritten by a special SMTP-time rewrite rule (see section 31.9), but no ordinary rewrite rules have yet been applied. If, however, the sender address is verified in the ACL, it is rewritten before verification, and remains rewritten thereafter. The subsequent value of `$sender_address` is the rewritten address. This also applies if sender verification happens in a RCPT ACL. Otherwise, when the sender address is not verified, it is rewritten as soon as a message's header lines have been received.

Similarly, at the start of an ACL for RCPT, the current recipient's address may have been rewritten by a special SMTP-time rewrite rule, but no ordinary rewrite rules have yet been applied to it. However, the behaviour is different from the sender address when a recipient is verified. The address is rewritten for the verification, but the rewriting is not remembered at this stage. The value of `$local_part` and `$domain` after verification are always the same as they were before (that is, they contain the unrewritten &ndash; except for SMTP-time rewriting &ndash; address).

As soon as a message's header lines have been received, all the envelope recipient addresses are permanently rewritten, and rewriting is also applied to the addresses in the header lines (if configured). This happens before adding

any header lines that were specified in MAIL or RCPT ACLs, and before the DATA ACL and local\_scan() functions are run.

When an address is being routed, either for delivery or for verification, rewriting is applied immediately to child addresses that are generated by redirection, unless no\_rewrite is set on the router.

At transport time, additional rewriting of addresses in header lines can be specified by setting the generic headers\_rewrite option on a transport. This option contains rules that are identical in form to those in the rewrite section of the configuration file. They are applied to the original message header lines and any that were added by ACLs or a system filter. They are not applied to header lines that are added by routers or the transport.

The outgoing envelope sender can be rewritten by means of the return\_path transport option. However, it is not possible to rewrite envelope recipients at transport time. 31.3 Testing the rewriting rules that apply on input

Exim's input rewriting configuration appears in a part of the run time configuration file headed by "begin rewrite". It can be tested by the -brw command line option. This takes an address (which can be a full RFC 2822 address) as its argument. The output is a list of how the address would be transformed by the rewriting rules for each of the different places it might appear in an incoming message, that is, for each different header and for the envelope sender and recipient fields. For example, `exim -brw ph10@exim.workshop.example`

```
might produce the output
sender: Philip.Hazel@exim.workshop.example
from: Philip.Hazel@exim.workshop.example
to: ph10@exim.workshop.example
cc: ph10@exim.workshop.example
bcc: ph10@exim.workshop.example
reply-to: Philip.Hazel@exim.workshop.example
env-from: Philip.Hazel@exim.workshop.example
env-to: ph10@exim.workshop.example
```

which shows that rewriting has been set up for that address when used in any of the source fields, but not when it appears as a recipient address. At the present time, there is no equivalent way of testing rewriting rules that are set for a particular transport. 31.4 Rewriting rules

The rewrite section of the configuration file consists of lines of rewriting rules in the form <source pattern> <replacement> <flags>

Rewriting rules that are specified for the headers\_rewrite generic transport option are given as a colon-separated list. Each item in the list takes the same form as a line in the main rewriting configuration (except that any colons must be doubled, of course).

The formats of source patterns and replacement strings are described below. Each is terminated by white space, unless enclosed in double quotes, in which case normal quoting conventions apply inside the quotes. The flags are single characters which may appear in any order. Spaces and tabs between them are ignored.

For each address that could potentially be rewritten, the rules are scanned in order, and replacements for the address from earlier rules can themselves be replaced by later rules (but see the "Q" and "R" flags).

The order in which addresses are rewritten is undefined, may change between releases, and must not be relied on, with one exception: when a message is received, the envelope sender is always rewritten first, before any header lines are rewritten. For example, the replacement string for a rewrite of an address in To: must not assume that the message's address in From: has (or has not) already been rewritten. However, a rewrite of From: may assume that the envelope sender has already been rewritten.

The variables \$local\_part and \$domain can be used in the replacement string to refer to the address that is being rewritten. Note that lookup-driven rewriting can be done by a rule of the form \*@\* \${lookup ...

where the lookup key uses \$1 and \$2 or \$local\_part and \$domain to refer to the address that is being rewritten. 31.5 Rewriting patterns

The source pattern in a rewriting rule is any item which may appear in an address list (see section 10.18). It is in fact processed as a single-item address list, which means that it is expanded before being tested against the address. As

always, if you use a regular expression as a pattern, you must take care to escape dollar and backslash characters, or use the \N facility to suppress string expansion within the regular expression.

Domains in patterns should be given in lower case. Local parts in patterns are case-sensitive. If you want to do case-insensitive matching of local parts, you can use a regular expression that starts with `^(?i)`.

After matching, the numerical variables \$1, \$2, etc. may be set, depending on the type of match which occurred. These can be used in the replacement string to insert portions of the incoming address. \$0 always refers to the complete incoming address. When a regular expression is used, the numerical variables are set from its capturing subexpressions. For other types of pattern they are set as follows:

-

If a local part or domain starts with an asterisk, the numerical variables refer to the character strings matched by asterisks, with \$1 associated with the first asterisk, and \$2 with the second, if present. For example, if the pattern `*queen@*.fict.example`

is matched against the address `hearts-queen@wonderland.fict.example` then \$0 = hearts-queen@wonderland.fict.example  
 \$1 = hearts-  
 \$2 = wonderland

Note that if the local part does not start with an asterisk, but the domain does, it is \$1 that contains the wild part of the domain.

-

If the domain part of the pattern is a partial lookup, the wild and fixed parts of the domain are placed in the next available numerical variables. Suppose, for example, that the address `foo@bar.baz.example` is processed by a rewriting rule of the form `*@partial-dbm;/some/dbm/file <replacement string>`

and the key in the file that matches the domain is `*.baz.example`. Then \$1 = foo  
 \$2 = bar  
 \$3 = baz.example

If the address `foo@baz.example` is looked up, this matches the same wildcard file entry, and in this case \$2 is set to the empty string, but \$3 is still set to `baz.example`. If a non-wild key is matched in a partial lookup, \$2 is again set to the empty string and \$3 is set to the whole domain. For non-partial domain lookups, no numerical variables are set.

31.6 Rewriting replacements

If the replacement string for a rule is a single asterisk, addresses that match the pattern and the flags are not rewritten, and no subsequent rewriting rules are scanned. For example, `hatta@lookingglass.fict.example * f`

specifies that `hatta@lookingglass.fict.example` is never to be rewritten in From: headers.

If the replacement string is not a single asterisk, it is expanded, and must yield a fully qualified address. Within the expansion, the variables `$local_part` and `$domain` refer to the address that is being rewritten. Any letters they contain retain their original case &ndash; they are not lower cased. The numerical variables are set up according to the type of pattern that matched the address, as described above. If the expansion is forced to fail by the presence of `&ldquo;fail&rdquo;` in a conditional or lookup item, rewriting by the current rule is abandoned, but subsequent rules may take effect. Any other expansion failure causes the entire rewriting operation to be abandoned, and an entry written to the panic log.

31.7 Rewriting flags

There are three different kinds of flag that may appear on rewriting rules:

-

Flags that specify which headers and envelope addresses to rewrite: E, F, T, b, c, f, h, r, s, t.

-

A flag that specifies rewriting at SMTP time: S.

Flags that control the rewriting process: Q, q, R, w.

For rules that are part of the headers\_rewrite generic transport option, E, F, T, and S are not permitted. 31.8 Flags specifying which headers and envelope addresses to rewrite

If none of the following flag letters, nor the &ldquo;S&rdquo; flag (see section 31.9) are present, a main rewriting rule applies to all headers and to both the sender and recipient fields of the envelope, whereas a transport-time rewriting rule just applies to all headers. Otherwise, the rewriting rule is skipped unless the relevant addresses are being processed.

E rewrite all envelope fields  
 F rewrite the envelope From field  
 T rewrite the envelope To field  
 b rewrite the Bcc: header  
 c rewrite the Cc: header  
 f rewrite the From: header  
 h rewrite all headers  
 r rewrite the Reply-To: header  
 s rewrite the Sender: header  
 t rewrite the To: header

You should be particularly careful about rewriting Sender: headers, and restrict this to special known cases in your own domains. 31.9 The SMTP-time rewriting flag

The rewrite flag &ldquo;S&rdquo; specifies a rewrite of incoming envelope addresses at SMTP time, as soon as an address is received in a MAIL or RCPT command, and before any other processing; even before syntax checking. The pattern is required to be a regular expression, and it is matched against the whole of the data for the command, including any surrounding angle brackets.

This form of rewrite rule allows for the handling of addresses that are not compliant with RFCs 2821 and 2822 (for example, &ldquo;bang paths&rdquo; in batched SMTP input). Because the input is not required to be a syntactically valid address, the variables \$local\_part and \$domain are not available during the expansion of the replacement string. The result of rewriting replaces the original address in the MAIL or RCPT command. 31.10 Flags controlling the rewriting process

There are four flags which control the way the rewriting process works. These take effect only when a rule is invoked, that is, when the address is of the correct type (matches the flags) and matches the pattern:

-

If the &ldquo;Q&rdquo; flag is set on a rule, the rewritten address is permitted to be an unqualified local part. It is qualified with qualify\_recipient. In the absence of &ldquo;Q&rdquo; the rewritten address must always include a domain.

-

If the &ldquo;q&rdquo; flag is set on a rule, no further rewriting rules are considered, even if no rewriting actually takes place because of a &ldquo;fail&rdquo; in the expansion. The &ldquo;q&rdquo; flag is not effective if the address is of the wrong type (does not match the flags) or does not match the pattern.

-

The &ldquo;R&rdquo; flag causes a successful rewriting rule to be re-applied to the new address, up to ten times. It can be combined with the &ldquo;q&rdquo; flag, to stop rewriting once it fails to match (after at least one successful rewrite).

-

When an address in a header is rewritten, the rewriting normally applies only to the working part of the address, with any comments and RFC 2822 &ldquo;phrase&rdquo; left unchanged. For example, rewriting might change From: Ford Prefect <fp42@restaurant.hitch.fict.example>

into From: Ford Prefect <prefectf@hitch.fict.example>

Sometimes there is a need to replace the whole address item, and this can be done by adding the flag letter &ldquo;w&rdquo; to a rule. If this is set on a rule that causes an address in a header line to be rewritten, the entire address is replaced, not just the working part. The replacement must be a complete RFC 2822 address, including the angle brackets if necessary. If text outside angle brackets contains a character whose value is greater than 126 or less than 32 (except for tab), the text is encoded according to RFC 2047. The character set is taken from headers\_charset, which defaults to ISO-8859-1.

When the `!w` flag is set on a rule that causes an envelope address to be rewritten, all but the working part of the replacement address is discarded. 31.11 Rewriting examples

Here is an example of the two common rewriting paradigms: `*@*.hitch.fict.example $1@hitch.fict.example`  
`*@hitch.fict.example ${lookup{$1}dbm{/etc/realnames}\`  
`{${value}fail}@hitch.fict.example bctfrF`

Note the use of `!fail` in the lookup expansion in the second rule, forcing the string expansion to fail if the lookup does not succeed. In this context it has the effect of leaving the original address unchanged, but Exim goes on to consider subsequent rewriting rules, if any, because the `!q` flag is not present in that rule. An alternative to `!fail` would be to supply `$1` explicitly, which would cause the rewritten address to be the same as before, at the cost of a small bit of processing. Not supplying either of these is an error, since the rewritten address would then contain no local part.

The first example above replaces the domain with a superior, more general domain. This may not be desirable for certain local parts. If the rule `root@*.hitch.fict.example *`

were inserted before the first rule, rewriting would be suppressed for the local part `root` at any domain ending in `hitch.fict.example`.

Rewriting can be made conditional on a number of tests, by making use of `$if` in the expansion item. For example, to apply a rewriting rule only to messages that originate outside the local host: `*@*.hitch.fict.example "$if !eq`  
`{${sender_host_address}}{\`  
`{$1@hitch.fict.example}fail}"`

The replacement string is quoted in this example because it contains white space.

Exim does not handle addresses in the form of `!bang paths`. If it sees such an address it treats it as an unqualified local part which it qualifies with the local qualification domain (if the source of the message is local or if the remote host is permitted to send unqualified addresses). Rewriting can sometimes be used to handle simple bang paths with a fixed number of components. For example, the rule `\N^([!]+)!(.*)@your.domain.example$N $2@$1` rewrites a two-component bang path `host.name!user` as the domain address `user@host.name`. However, there is a security implication in using this as a global rewriting rule for envelope addresses. It can provide a backdoor method for using your system as a relay, because the incoming addresses appear to be local. If the bang path addresses are received via SMTP, it is safer to use the `!S` flag to rewrite them as they are received, so that relay checking can be done on the rewritten addresses.