NAME

regexp_table - format of Postfix regular expression tables

SYNOPSIS

postmap -q "*string***" regexp:/etc/postfix/***filename*

postmap -q - regexp:/etc/postfix/*filename <inputfile*

DESCRIPTION

The Postfix mail system uses optional tables for address rewriting, mail routing, or access control. These tables are usually in **dbm** or **db** format.

Alternatively, lookup tables can be specified in POSIX regular expression form. In this case, each input is compared against a list of patterns. When a match is found, the corresponding result is returned and the search is terminated.

To find out what types of lookup tables your Postfix system supports use the "**postconf -m**" command.

To test lookup tables, use the "**postmap -q**" command as described in the SYNOPSIS above. Use "**postmap -hmq -** <*file*" for header_checks(5) patterns, and "**postmap -bmq -** <*file*" for body_checks(5) (Postfix 2.6 and later).

COMPATIBILITY

With Postfix version 2.2 and earlier specify "**postmap -fq**" to query a table that contains case sensitive patterns. Patterns are case insensitive by default.

TABLE FORMAT

The general form of a Postfix regular expression table is:

/pattern/flags result

When *pattern* matches the input string, use the corresponding *result* value.

!*/pattern/flags result*

When *pattern* does **not** match the input string, use the corresponding *result* value.

if */pattern/flags*

endif If the input string matches */pattern/*, then match that input string against the patterns between **if** and **endif**. The **if**..**endif** can nest.

Note: do not prepend whitespace to patterns inside **if**..**endif**.

This feature is available in Postfix 2.1 and later.

if !*/pattern/flags*

endif If the input string does not match */pattern/*, then match that input string against the patterns between **if** and **endif**. The **if**..**endif** can nest.

Note: do not prepend whitespace to patterns inside **if**..**endif**.

This feature is available in Postfix 2.1 and later.

blank lines and comments

Empty lines and whitespace-only lines are ignored, as are lines whose first non-whitespace character is a '#'.

multi-line text

A logical line starts with non-whitespace text. A line that starts with whitespace continues a logical line.

Each pattern is a POSIX regular expression enclosed by a pair of delimiters. The regular expression syntax is documented in **re_format(7)** with 4.4BSD, in **regex(5)** with Solaris, and in **regex(7)** with Linux. Other systems may use other document names.

The expression delimiter can be any non-alphanumerical character, except whitespace or characters that have special meaning (traditionally the forward slash is used). The regular expression can contain whitespace.

By default, matching is case-insensitive, and newlines are not treated as special characters. The behavior is controlled by flags, which are toggled by appending one or more of the following characters after the pattern:

i (default: on)
: Toggles the case sensitivity flag. By default, matching is case insensitive.

m (default: off)
: Toggle the multi-line mode flag. When this flag is on, the **^** and **$** metacharacters match immediately after and immediately before a newline character, respectively, in addition to matching at the start and end of the input string.

x (default: on)
: Toggles the extended expression syntax flag. By default, support for extended expression syntax is enabled.

TABLE SEARCH ORDER

Patterns are applied in the order as specified in the table, until a pattern is found that matches the input string.

Each pattern is applied to the entire input string. Depending on the application, that string is an entire client hostname, an entire client IP address, or an entire mail address. Thus, no parent domain or parent network search is done, and *user@domain* mail addresses are not broken up into their *user* and *domain* constituent parts, nor is *user+foo* broken up into *user* and *foo*.

TEXT SUBSTITUTION

Substitution of substrings (text that matches patterns inside "()") from the matched expression into the result string is requested with $1, $2, etc.; specify $$ to produce a $ character as output. The macros in the result string may need to be written as ${n} or $(n) if they aren't followed by whitespace.

Note: since negated patterns (those preceded by **!**) return a result when the expression does not match, substitutions are not available for negated patterns.

EXAMPLE SMTPD ACCESS MAP

```
# Disallow sender-specified routing. This is a must if you relay mail
# for other domains.
/[%!@].*[%!@]/ 550 Sender-specified routing rejected

# Postmaster is OK, that way they can talk to us about how to fix
# their problem.
/^postmaster@/ OK

# Protect your outgoing majordomo exploders
if !/^owner-/
/^(.*)-outgoing@(.*)$/ 550 Use ${1}@${2} instead
endif
```

EXAMPLE HEADER FILTER MAP

```
# These were once common in junk mail.
/^Subject: make money fast/ REJECT
/^To: friend@public\.com/ REJECT
```

EXAMPLE BODY FILTER MAP

```
# First skip over base 64 encoded text to save CPU cycles.
~^[[:alnum:]+/]{60,}$~ OK

# Put your own body patterns here.
```

SEE ALSO

postmap(1),
Postfix lookup table manager

pcre_table(5), format of PCRE tables
cidr_table(5),
format of CIDR tables

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
DATABASE_README, Postfix lookup table overview

AUTHOR(S)

The regexp table lookup code was originally written by:
LaMont Jones
lamont@hp.com

That code was based on the PCRE dictionary contributed by:
Andrew McNamara
andrewm@connect.com.au
connect.com.au Pty. Ltd.
Level 3, 213 Miller St
North Sydney, NSW, Australia

Adopted and adapted by:
Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

Wietse Venema
Google, Inc.
111 8th Avenue
New York, NY 10011, USA