NAME

pgsql_table − Postfix PostgreSQL client configuration

SYNOPSIS

postmap −q "*string***" pgsql:/etc/postfix/***filename*

postmap −q − pgsql:/etc/postfix/*filename* <*inputfile*

DESCRIPTION

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

Alternatively, lookup tables can be specified as PostgreSQL databases. In order to use PostgreSQL lookups, define a PostgreSQL source as a lookup table in main.cf, for example:

alias_maps = pgsql:/etc/pgsql−aliases.cf

The file /etc/postfix/pgsql−aliases.cf has the same format as the Postfix main.cf file, and can specify the parameters described below.

BACKWARDS COMPATIBILITY

For compatibility with other Postfix lookup tables, PostgreSQL parameters can also be defined in main.cf. In order to do that, specify as PostgreSQL source a name that doesn't begin with a slash or a dot. The PostgreSQL parameters will then be accessible as the name you've given the source in its definition, an underscore, and the name of the parameter. For example, if the map is specified as "pgsql:*pgsqlname*", the parameter "hosts" below would be defined in main.cf as "*pgsqlname*_hosts".

Note: with this form, the passwords for the PostgreSQL sources are written in main.cf, which is normally world−readable. Support for this form will be removed in a future Postfix version.

Normally, the SQL query is specified via a single **query** parameter (described in more detail below). When this parameter is not specified in the map definition, Postfix reverts to an older interface, with the SQL query constructed from the **select_function**, **select_field**, **table**, **where_field** and **additional_conditions** parameters. The old interface will be gradually phased out. To migrate to the new interface set:

query = SELECT *select_function*('%s')

or in the absence of **select_function**, the lower precedence:

query = SELECT *select_field*
FROM *table*
WHERE *where_field* = '%s'
additional_conditions

Use the value, not the name, of each legacy parameter. Note that the **additional_conditions** parameter is optional and if not empty, will always start with **AND**.

LIST MEMBERSHIP

When using SQL to store lists such as $mynetworks, $mydestination, $relay_domains, $local_recipient_maps, etc., it is important to understand that the table must store each list member as a separate key. The table lookup verifies the *existence* of the key. See "Postfix lists versus tables" in the DATABASE_README document for a discussion.

Do NOT create tables that return the full list of domains in $mydestination or $relay_domains etc., or IP addresses in $mynetworks.

DO create tables with each matching item as a key and with an arbitrary value. With SQL databases it is not uncommon to return the key itself or a constant value.

PGSQL PARAMETERS

hosts The hosts that Postfix will try to connect to and query from. Specify *unix:* for UNIX−domain sockets, *inet:* for TCP connections (default). Example:

hosts = host1.some.domain host2.some.domain:port
hosts = unix:/file/name

The hosts are tried in random order, with all connections over UNIX domain sockets being tried before those over TCP. The connections are automatically closed after being idle for about 1 minute, and are re−opened as necessary.

NOTE: the *unix:* and *inet:* prefixes are accepted for backwards compatibility reasons, but are actually ignored. The PostgreSQL client library will always try to connect to an UNIX socket if the name starts with a slash, and will try a TCP connection otherwise.

user, password

The user name and password to log into the pgsql server. Example:

user = someone
password = some_password

dbname

The database name on the servers. Example:

dbname = customer_database

query The SQL query template used to search the database, where **%s** is a substitute for the address Postfix is trying to resolve, e.g.

query = SELECT replacement FROM aliases WHERE mailbox = '%s'

This parameter supports the following '%' expansions:

%% This is replaced by a literal '%' character. (Postfix 2.2 and later)

%s This is replaced by the input key. SQL quoting is used to make sure that the input key does not add unexpected metacharacters.

%u When the input key is an address of the form user@domain, **%u** is replaced by the SQL quoted local part of the address. Otherwise, **%u** is replaced by the entire search string. If the localpart is empty, the query is suppressed and returns no results.

%d When the input key is an address of the form user@domain, **%d** is replaced by the SQL quoted domain part of the address. Otherwise, the query is suppressed and returns no results.

%[SUD]

The upper−case equivalents of the above expansions behave in the **query** parameter identically to their lower−case counter−parts. With the **result_format** parameter (see below), they expand the input key rather than the result value.

The above %S, %U and %D expansions are available with Postfix 2.2 and later

%[1−9]

The patterns %1, %2, ... %9 are replaced by the corresponding most significant component of the input key's domain. If the input key is *user@mail.example.com*, then %1 is **com**, %2 is **example** and %3 is **mail**. If the input key is unqualified or does not have enough domain components to satisfy all the specified patterns, the query is suppressed and returns no results.

The above %1, ... %9 expansions are available with Postfix 2.2 and later

The **domain** parameter described below limits the input keys to addresses in matching domains. When the **domain** parameter is non−empty, SQL queries for unqualified addresses or addresses in non−matching domains are suppressed and return no results.

The precedence of this parameter has changed with Postfix 2.2, in prior releases the precedence was, from highest to lowest, **select_function**, **query**, **select_field**, ...

With Postfix 2.2 the **query** parameter has highest precedence, see COMPATIBILITY above.

NOTE: DO NOT put quotes around the **query** parameter.

result_format (default: %s)

Format template applied to result attributes. Most commonly used to append (or prepend) text to the result. This parameter supports the following '%' expansions:

%% This is replaced by a literal '%' character.

%s This is replaced by the value of the result attribute. When result is empty it is skipped.

%u When the result attribute value is an address of the form user@domain, **%u** is replaced by the local part of the address. When the result has an empty localpart it is skipped.

%d When a result attribute value is an address of the form user@domain, **%d** is replaced by the domain part of the attribute value. When the result is unqualified it is skipped.

%[SUD1−9]

The upper−case and decimal digit expansions interpolate the parts of the input key rather than the result. Their behavior is identical to that described with **query**, and in fact because the input key is known in advance, queries whose key does not contain all the information specified in the result template are suppressed and return no results.

For example, using "result_format = smtp:[%s]" allows one to use a mailHost attribute as the basis of a transport(5) table. After applying the result format, multiple values are concatenated as comma separated strings. The expansion_limit and parameter explained below allows one to restrict the number of values in the result, which is especially useful for maps that must return at most one value.

The default value **%s** specifies that each result value should be used as is.

This parameter is available with Postfix 2.2 and later.

NOTE: DO NOT put quotes around the result format!

domain (default: no domain list)

This is a list of domain names, paths to files, or dictionaries. When specified, only fully qualified search keys with a *non−empty* localpart and a matching domain are eligible for lookup: 'user' lookups, bare domain lookups and "@domain" lookups are not performed. This can significantly reduce the query load on the PostgreSQL server.

domain = postfix.org, hash:/etc/postfix/searchdomains

It is best not to use SQL to store the domains eligible for SQL lookups.

This parameter is available with Postfix 2.2 and later.

NOTE: DO NOT define this parameter for local(8) aliases, because the input keys are always unqualified.

expansion_limit (default: 0)

A limit on the total number of result elements returned (as a comma separated list) by a lookup against the map. A setting of zero disables the limit. Lookups fail with a temporary error if the limit is exceeded. Setting the limit to 1 ensures that lookups do not return multiple values.

OBSOLETE QUERY INTERFACES

This section describes query interfaces that are deprecated as of Postfix 2.2. Please migrate to the new **query** interface as the old interfaces are slated to be phased out.

select_function

This parameter specifies a database function name. Example:

select_function = my_lookup_user_alias

This is equivalent to:

query = SELECT my_lookup_user_alias('%s')

This parameter overrides the legacy table−related fields (described below). With Postfix versions prior to 2.2, it also overrides the **query** parameter. Starting with Postfix 2.2, the **query** parameter has highest precedence, and the **select_function** parameter is deprecated.

The following parameters (with lower precedence than the **select_function** interface described above) can be used to build the SQL select statement as follows:

SELECT [**select_field**]
FROM [**table**]
WHERE [**where_field**] = '%s'
[**additional_conditions**]

The specifier %s is replaced with each lookup by the lookup key and is escaped so if it contains single quotes or other odd characters, it will not cause a parse error, or worse, a security problem.

Starting with Postfix 2.2, this interface is obsoleted by the more general **query** interface described above. If higher precedence the **query** or **select_function** parameters described above are defined, the parameters described here are ignored.

select_field

The SQL "select" parameter. Example:

select_field = forw_addr

table The SQL "select .. from" table name. Example:

table = mxaliases

where_field

The SQL "select .. where" parameter. Example:

where_field = alias

additional_conditions

Additional conditions to the SQL query. Example:

additional_conditions = AND status = 'paid'

SEE ALSO

postmap(1), Postfix lookup table manager
postconf(5), configuration parameters
ldap_table(5), LDAP lookup tables
mysql_table(5), MySQL lookup tables
sqlite_table(5), SQLite lookup tables

README FILES

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

LICENSE

HISTORY

PgSQL support was introduced with Postfi x version 2.1.

AUTHOR(S)

Based on the MySQL client by:
Scott Cotton, Joshua Marcus
IC Group, Inc.

Ported to PostgreSQL by:
Aaron Sethman

Further enhanced by:
Liviu Daia
Institute of Mathematics of the Romanian Academy
P.O. BOX 1−764
RO−014700 Bucharest, ROMANIA