PowerMail manual

PowerDNS BV

pdns@powerdns.com

PowerMail manual
by

Published v1.0 $Date: 2002/08/30 13:04:55 $

Explanation of PowerMail concepts, installation and operation.

Table of Contents

Chapter 1. PowerMail introduction

PowerMail is a redundant & distributed system for receiving mail and storing it for users. The way PowerMail works is quite unorthodox, this document sets out how the different modules cooperate.

PowerMail only receives email and makes it available over POP. Furthermore, incoming email can be forwarded to other email addresses. Although most PowerMail installations will query a relational database for information about email addresses, messages are stored on disk.

The real strength of PowerMail lies in:

- One single source of information about mailboxes. No /etc/passwd, no 'recipientdomains', or whatever. If a mailbox is listed in the userbase, it works.
- Ability to scale. Distributed message hosting means that additional machines can be added on the fly without moving files around or growing filesystems.
- Redundancy. Besides distributing messages over multiple machines, additional copies can be stored, leading to very high redundancy. Furthermore, it is easily possible to have multiple pop servers online at once, as well as multiple smtp servers, all talking to the same pool of messages. There is no single point of failure. And no NFS either.
- Very high performance. PowerMail is about receving email. There is no queue, email is never bounced. Either email is accepted or it is not - there is no way for the system to get jammed with incoming email. Messages are sent straight to the disk. Retrieval works the same way, in reverse. On operating systems supporting this, `sendfile()` zerocopy TCP features are exploited for maximum throughput.

It is best to not compare PowerMail to traditional mailhosting solutions as this can be very confusing. Specifically:

- There is no mapping between mailboxes and UNIX users. All user data is always read from the User Base, which is typically a relational database
- Mailbox centric and not domain centric. A mailbox exists or it doesn't. There is no reason to configure which domains PowerMail is to host mailboxes for.
- SMTP and POP daemons can be on different machines than the messages. PowerSMTP and PowerPOP do not access the disk directly - they contact the backends for that purpose.
- Messages can live on multiple machines simultaneously and redundantly. A mailbox may span multiple machines.

PowerMail can best be thought of as mail hosting where adding a new mailbox consists of a single SQL insert statement.

1.1. Concepts

The following concepts are used by PowerMail, followed by their specific meaning within PowerMail:

Mailboxes

PowerMail contains an arbitrary number of Mailboxes. Contrary to existing practice, a mailbox name includes the domain name. A sample mailbox might be called 'info@company.biz'. PowerMail does not care which domain a mailbox resides under - the mailbox name includes the domain.

Userbase

To know if a mailbox exists, the Userbase is queried. This will most often be a relational database. The stock PowerMail distribution works with MySQL. Besides storing which mailboxes exist, the Userbase also knows about Quotas and Mailforwarding.

Messages

A mailbox consists of messages. Each message is a single file on disk, even if it has multiple recipients. In other words, a single file can live in multiple mailboxes simultaneously. This means that a message with thousands of recipients is stored only once on disk.

Backends

The mentioned messages live on backends. Each backend is a very simpleminded fileserver. Typically, a single backend is used per host, unless that host has multiple filesystems, in which case a backend is started for each filesystem. Backends can be additively configured. A backend is accessed via the **pplistener** command.

Backends store messages as files on disk. Each message is one file.

Redundancy & Distributed Hosting

Each message may live on multiple backends. This can be used both for redundancy as well as for bandwidth purposes. It is possible to assign storage backends to specific locations and make sure that POP traffic thus remains local.

Daemons

Mail is received over the Simple Mail Tranport Protocol, SMTP. It is transferred to the reader via the Post Office Protocol, version 3, POP3.

The powersmtp daemon

The powersmtp daemon listens on the SMTP port (25) and checks with the Userbase if recipients exist. If they do, the message is received and sent to a backend. The recipient might also be a forwarded account, in which case the message is passed to the outgoing SMTP host.

The powerpop daemon

The powerpop daemon listens on the pop3 port (110) and checks with the Userbase if the supplied user and password data exist in the database. If this is the case, all messages for this mailbox are retrieved from the backends. Duplicates are filtered out and the unified list is presented to the user.

The pplistener daemon

Each backend runs the pplistener daemon which can be likened to a very simple ftp server. Each pplistener can be in read/write or in readonly mode. The latter might for example occur if the filesystem is full, or if the backend is about to be phased out and should accept no new data.

1.2. What PowerMail is not, roadmap

PowerMail only receives email. Sending email is a well solved problem. When email needs to be forwarded, PowerMail needs help from an outgoing mailserver.

Furthermore, PowerMail will not satisfy the demanding UNIX user. There is no procmail support, for example. PowerMail is a low-cost bulk email hosting solution. If more functionality is needed within domains served by PowerMail, the use of email forwarding is suggested. For example, the domain **powerdns.com** is hosted on PowerMail. Yet <`support@powerdns.com`> is a Mailman mailinglist. This is achieved by forwarding <`support@powerdns.com`> to <`support@mailman.powerdns.com`>.

Currently, there is also no IMAP support but that will change soon.

Furthermore, it is also expected that support will be added for piping mail through programs like Amavis and SpamAssassin to mitigate viruses and spam.

Chapter 2. Release notes

Before proceeding, you should check the release notes for your PowerMail version, as specified in the name of the distribution file.

2.1. 1.4

This version now runs on the toplevel domain .TK, replacing the earlier legacy email forwarding system. A lot of work has been done to handle the 1000+ concurrent connections sometimes needed to handle .TK traffic.

Furthermore, the Oracle backend (available in source on request) now reads its query from the configuration, allowing it to work with many different schemas.

2.2. 1.3.1

Important upgrade. This version had the ability to saturate a MySQL server with connections under certain circumstances. PostgreSQL users are not affected by this bug.

- The PowerMail MySQL driver neglected to call mysql_close() on discarded connections beyond the standard 2 allocated ones.

2.3. 1.3

This release fixes some RFC violations and implements some of the optional bits.

- RFC Optional length output added to 'RETR', allowing older Netscape versions to display a progress bar while downloading. Thanks to Peter van Dijk.
- Added STAT output to PASS which is appreciated by some POP3 clients. Thanks to Peter van Dijk.
- RFC violation: STAT used to count already deleted messages. Thanks to Peter van Dijk.
- RFC violation: TOP x 0 displayed entire message instead of only headers. This would have caused slow behaviour on clients only wishing to see headers. Thanks to Peter van Dijk.
- Tiny improvements in logging rationale.

2.4. 1.2.0

This is a big release. We salute Julian Seward for his wonderful work on the Valgrind memory debugger (http://developer.kde.org/~sewardj/). PowerMail is now a better product for it.

New features:

- Now spawns multiple database connections if those are needed using a connection pool.
- Added PostgreSQL support, see Section 10.4.
- Added catch-all/fallback accounts, see Section 9.3.
- Added delay after a wrong password to prevent dictionary attacks.
- Added missing **pptool** documentation, see Section 7.1.

Bugs fixed:

- Removed memory leaks, thanks to Valgrind
- Increased stability by removing duplicate deletes, thanks to Valgrind
- **pplistener** now logs under the right program name
- **pplistener** did not allow connections to be closed gracefully and logged a lot of unneeded 'connection reset by peer' messages as a result.

Works in progress:

- FreeBSD version - nearly done.

2.5. 1.1.0

This release adds crypted password support, bringing the number of password schemes to three: plaintext, crypt and md5. See Section 9.1 for details and important information for deployments with plaintext passwords starting with a '{' and containing a '}' further on!

- Improved SMTP error codes sent out to convince remote MTAs that non-existant accounts really don't exist.

- {crypt} and {md5} support, plaintext passwords should now preferrably be prefixed with {crypt}. See Section 9.1 for important upgrade information!
- Added some convenience functions to **pptool** to help calculate md5 hashes and crypts. See Section 7.1.1.
- Simplified internal userbase interface to ease integration of LDAP and Oracle backends.

2.6. 1.0.0

First public release of PowerMail. Changes since less widely released versions:

- RPM neglected to create `/var/powermail`. Thanks to Dave Aaldering for noticing this.
- init.d scripts now contain hints for RedHat's **ntsysv** tool. Thanks again to Dave Aaldering.

Chapter 3. Advanced users quick installation

This is for the people that don’t do documentation and are ready to deploy now.

If you have a mailserver running already, move PowerMail to another IP address. To do so, edit the files in `/etc/powermail`.

Now start the daemons:

```
# /etc/init.d/pplistener start
pplistener: started

# /etc/init.d/pplistener status
        pplistener: pplistener functioning ok: +OK OK!

# /etc/init.d/powersmtp start
        powersmtp: started

# /etc/init.d/powersmtp status
        powersmtp: PowerSMTP functioning ok: 200 OK!

# /etc/init.d/powerpop start
powerpop: started

# /etc/init.d/powerpop status
powerpop: PowerPOP functioning ok: +OK OK!
```

To check if all is well, run pptool:

```
# pptool status
    127.0.0.1   223 mb, 452547 inodes, load: 0.15, read/write access
                        223 mb available for writing
```

This means that mail storage is online and functioning. Now check if the database connection is functioning:

```
# pptool orphans
        No orphans!
```

This verifies that a connection could be made to the ’userbase’. More about orphans and the userbase later. Now deliver yourself a message. Depending on your setup, you may choose to deliver it by hand over SMTP or to move an MX record to your actual server, and use a regular mailing agent. After delivering a message, we can execute pptool again:

```
$ pptool dir
127.0.0.1   1 mailbox

Mailbox              # backends
info@example.com  1

$ pptool list info@example.com
127.0.0.1   done listing ’info@example.com’, 1 messages

1022754059:211363:snapcount.0.1
  127.0.0.1:1101 157
   * NOT REDUNDANT!

1 messages, 100% non-redundant
0 kilobytes net
```

To retrieve the message, connect to the popserver and login as ’info@example.com’, and the message will be there.

To add users, edit `/etc/powermail/mailboxes`. There is no need to inform PowerMail of your changes, they are picked up automatically.

For more details, see the next chapter. If everything is clear already, head on to Section 10.5 and Chapter 6. Otherwise read the rest of the documentation for further explanation.

Chapter 4. Starting a simple MySQL installation

This is a more elaborate version of the previous chapter, this time connecting to MySQL. To learn PowerMail operations, it is suggested to start with a simple installation.

4.1. Backend hosts: pplistener

The pplistener actually stores and serves messages to users. There might be many pplisteners but in our simple installation there will be only one. The backend daemon pplistener defaults to using /var/powermail for storing messages, you may whish to change this in `/etc/powermail/pplistener.conf`.

Now start pplistener:

```
# /etc/init.d/pplistener start
pplistener: started
# /etc/init.d/pplistener status
pplistener: pplistener functioning ok: +OK OK!
```

4.2. Internet daemons: powerpop & powersmtp

The most tricky part of these daemons is setting up the communication with the userbase, implemented by default as a MySQL database. This userbase is called ’mysqlplain’ and is documented Section 10.1.

4.2.1. Setting up MySQL

Connect to MySQL as a user that can create database, and issue the following statements:

```
$ mysql -u root
mysql> create database powermail;
mysql> use powermail;
mysql> CREATE TABLE mboxes (
id int(11) NOT NULL auto_increment,
mbox varchar(128) NOT NULL,
password varchar(20) NOT NULL,
quotaMB int(11) default ’0’,
isForward tinyint(1) default ’0’,
fwdDest varchar(80) default NULL,
PRIMARY KEY  (id),
UNIQUE KEY mbox_index (mbox)
);
mysql> INSERT INTO mboxes (mbox,password) VALUES (’info@example.com’,’s3cr3t!’);
```

In this simple example, we will not mention mysql authentication but in real life you should use 'GRANT' statements to secure readonly access to your database.

Note that you also generated a mailbox called info@example.com, with password 's3cr3t!'.

4.2.2. Configuring

Both powersmtp and powerpop need to be able to connect to MySQL. Because of these shared parameters, both read the file `power.conf` before parsing their own configuration files.

In power.conf, specify your 'mysql-host', 'mysql-user', and 'mysql-password'. Also set 'userbase' to 'mysqlplain' to use this MySQL database.

4.2.3. Choosing IP addresses & ports to bind to

You are now mostly ready to launch the daemons, but it is likely that a mailer is already running. This is not a problem but another port or IP address for running PowerMail must be chosen. Alternatively, stop your existing mailer.

If you decide to use a separate IP address, create an interface alias for it. This will differ with your operating system. Next tell PowerMail about it. Again, the power.conf file is a apropriate file, as all daemons will bind to your IP address. A sample line might read 'listen-address=a.b.c.d', where a.b.c.d is your IP alias.

Another solution is to bind to another port, which is probably only useful for testing. As all daemons need a different port, specify this in powersmtp.conf and powerpop.conf using the listen-port parameter.

4.2.4. Launching

```
# /etc/init.d/powersmtp start
powersmtp: started
# /etc/init.d/powerpop start
powerpop:started
# /etc/init.d/powerpop status
powerpop: PowerPOP functioning ok: +OK OK!
# /etc/init.d/powersmtp status
powersmtp: PowerSMTP functioning ok: 200 OK!
```

If anything is wrong, most likely with the MySQL connection, these scripts will warn you, but launch nonetheless. This is because a temporary database failure should not shutdown your mailsystem permanently.

4.2.5. Testing

Now run the 'pptool' command, installed by default in /usr/bin:

```
# pptool stat
127.0.0.1   3792 mb, 568720 inodes, load: 0.01, read/write access
```

If there are any problems, for example with wrong passwords, this tool will inform you of them.

Next, we will test a sample delivery and retrieval of a message. First delivery, by hand, using the telnet tool. We will assume that powersmtp is running on port 25, but you might have chosen a different port for testing:

```
$ telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 snapcount PowerSMTP ESMTP
mail from: ahu@ds9a.nl
250 Started message <1006724301:884450:snapcount.0> for sender <ahu@ds9a.nl>
rcpt to: info@yourdomain.com
250 Added recipient 'info@yourdomain.com' to message <1006724301:884450:snapcount.0>
data
354 Enter message, ending with '.' on a line by itself. Quota available: 50000 kilobytes
From: ahu@ds9a.nl
To: info@yourdomain.com
Subject: test

testing 1 2 3
.
250 Delivered message <1006724301:884450:snapcount.0> successfully to all recipients
quit
221 bye
Connection closed by foreign host.
```

Now let's see if this message has arrived with the 'pptool' command:

```
# pptool list info@yourdomain.com
127.0.0.1        done listing 'info@yourdomain.com', 1 messages
```

```
1006724301:884450:snapcount.0.1
127.0.0.1               162
1.2.3.4                 162

1 messages, 0% non-redundant
0 kilobytes net
```

Next, we'll try to actually retrieve the message:

```
# telnet 127.0.0.1 110
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
+OK snapcount
user info@yourdomain.com
+OK
pass 31337pw
+OK
list
+OK
1 162
.
retr 1
+OK
Received: from 127.0.0.1:36724 by snapcount (PowerMail) with id <1006724301:884450:snapcoun
From: ahu
To: ahu@powerdns.org
Subject: test

testing 1 2 3
.
quit
+OK
Connection closed by foreign host.
```

Chapter 5. Configuration

PowerSMTP and PowerPOP share many common configuration items. Prime among these are the userbase connection details and possible **pplistener** backend password details.

To enable easy configuration, both PowerSMTP and PowerPOP first parse `power.conf`, followed by `powersmtp.conf` and `powerpop.conf` respectively.

PPListener is different and does not need userbase configuration and therefore does not parse `power.conf` but only `pplistener.conf`.

5.1. All settings

Common to PowerSMTP and PowerPOP:

backends

: Which backends to use for storing and retrieving messages. Syntax: group:priority:host. See Chapter 6 for details.

pplistener-password

: Password to use when authenticating with PPListener backends.

config-name

: Name of this virtual configuration. FIXME: write chapter on virtual configurations

listen-address

: IP address to bind to. Defaults to 0.0.0.0 which means 'all interfaces'.

listen-port

: Port to bind to. Defaults to 25 for PowerSMTP and to 110 for PowerPOP.

max-userbase-connections

: Maximum number of simultaneous database connections to open. Multiple connections share a single connection, so there is no need to set this very high. Defaults to 3.

max-userbase-spares

: A few userbase connections are kept on hand, even if they are not needed. This specifies how many spares can exist. If set to 0, connections will be opened and closed very often.

userbase

: Which userbase to query for mailbox data. Defaults to 'mysqlplain'. See Chapter 10.

run-as-gid
run-as-uid

Credentials to switch to after acquiring priviledged ports.

daemon=[yes|no]

To run as a daemon or not. ’no’ can be useful for debugging purposes.

PowerPOP and PowerSMTP also share userbase connection details for which see Chapter 10.

Chapter 6. Moving beyond a single host

The true power of PowerMail only comes to light when employing multiple machines. To access multiple storage backends (which each run **pptalker**), these need to be defined in `power.conf`. To do so, define the 'backends' configuration field, which lists each backend and which 'group' it belongs to aswell as its priority.

Each backend is listed according to the following syntax: 'group:priority:ipaddress'. Below a list of sample configurations and what they mean:

backends=1:1:192.168.0.1, 2:1:192.168.0.2

Mail is stored over 192.168.0.1 and 192.168.0.2. If redundancy-target is set to 2, both backends will always receive a copy of each message. If this target isn't met, email reception is denied. Each server is in its own group, which tells PowerMail that the two hosts are mutually redundant.

If `redundancy-target` is set to 1, the default, email will be sent to the backend with the most free disk space available. Alternatively, if `spread-load` is specified, it will be sent to the host with the lowest load average.

backends=1:1:192.168.0.1, 1:1:192.168.0.2, 1:2:192.168.100.100

Mail is stored on either 192.168.0.1 or 192.168.0.2, according to the `spread-load` or `spread-disk` specification above. If both of these are unavailable, mail is stored on 192.168.100.100.

backends=1:1:192.168.0.1, 2:1:192.168.0.2, 2:2:192.168.100.100

With `redundancy-target=2`, mail is normally stored on 192.168.0.1 and 192.168.0.2. If 192.168.0.2 is down, it will be stored on 192.168.0.1 and 192.168.100.100.

When not operating over localhost (127.0.0.1) configuring a password is mandatory. The password must be identical over all nodes in your PowerMail constellation. It is configured in `power.conf` and `pplistener.conf`.

This is advised even when running on localhost!

Chapter 7. Control and insight into powermail

7.1. pptool

A large powermail installation might well span tens of hosts and see thousands of mailboxes created and deleted each week. To cope with this, the program **pptool** has been provided.

As **pptool** also reads the common configuration file, it is aware of all backend storage pptalkers available, and can list or modify their contents.

The following commands are available:

status

Lists the status of all pptalkers, including the amount of free diskspace and inodes, the load average and importantly, the read/write status. It also lists the combined amount of diskspace available for writing.

dir [backend1] [backend2]

Lists all mailboxes in all backends. Specific backends can also be listed and only this will be queried then.

orphans

Lists all mailboxes that do exist on backends, but have no entry in the userbase. These mailboxes are ’orphaned’ and were not cleaned up properly.

purge

Purges a mailbox, wether it is in the database or not.

purge-orphans

Creates a list of orphans and purges them. Dangerous command!

list mailbox

Lists the message ids in a mailbox and on which backends it is stored, and if redundancy is met

block backend

Sets a backend to blocked, which means that it is administratively read only. No new mail will be accepted at that backend. Useful for phasing out servers, as deletes will be applied. Leaving a backend blocked will slowly empty it as users retrieve and delete mail.

unblock backend

Reverse operation.

usage mailbox

Prints the quota, in kilobytes, of a mailbox followed by its actual usage, separated by a colon. Suitable for machine parsing.

7.1.1. Convenience commands (password hashing)

Besides controling PowerMail nodes, **pptool** can also be used to calculate UNIX crypt()s and MD5 hashes for use in userbases to store passwords as described in Section 9.1.

crypt password

Prints the UNIX crypt(3) of the password supplied on the commandline.

md5 password

Prints the UNIX md5 hash of the password supplied on the commandline.

7.2. mboxdir

To find out where an actual mailbox lives on disk, the **mboxdir** command is provided. It accepts a single email address as a parameter and outputs the calculated directory. Please note that it will always have output - the userbase is not contacted to determine if this mailbox is actually present in PowerMail.

This is actually a feature as it allows the operator to find orphaned mailboxes. It is useful to do the following:

```
~# cd $(mboxdir info@example.com)
/var/messages/78/34/info@example.com# ls
cur  tmp
/var/messages/78/34/info@example.com#
```

Chapter 8. Redundancy

PowerSmtp makes sure that the redundancy target is met before announcing to the sending mailserver that the message was properly received. In this way, messages are always stored at at least the configured minimum redundancy.

group

Each backend is assigned to a group. Backends within a group do not generate redundancy - this can be used to make sure that email is spread over different physical locations so that each site has a copy of all articles.

Multiple partitions or filesystems residing withing a same host are also candidates to reside in the same group.

Priority

Data is always stored on the highest-priority available host. By assigning different priorities, it is possible to have 'hot spare' servers which can instantly take the place of failed or replaced boxes.

Load & diskspace

Given equal priorities, powersmtp can be instructed to choose a random server, or prefer the one with the most diskspace available or the one with the lowest load.

Your choice will depend on the workload presented.

8.1. Goals

It is important to determine what targets we want to meet. PowerMail tries to maximize perceived uptime - users must at all times be able to access their mailboxes, which should contain as much of their mail as possible. Access to POP is never disabled.

Mail delivery however is halted as soon as the minimum configured redundancy cannot be guaranteed, remote mailers will have to wait until more backends become available. As this is not directly perceived by the user, disabling deliveries is preferred over allowing non-redundant deliveries.

8.2. Redundancy in practice

When configured to operate in redundant mode, PowerMail only considers a message delivered when it is stored on more than 2 servers residing in different groups, and that number can be raised. If less than 2 servers are available, powersmtp starts to refuse mail delivery, giving out temporary ’please try later’ errors. This puts the burden of queueing on the delivering agent, where it belongs.

When retrieving mail, all backends are queried for which messages they have listed for that mailbox. In case of drastic failure, messages might start to disappear from a mailbox and return only when backends a restored.

8.2.1. Queued deletions

A backend that has been out of service might on return cause messages that the user has deleted to inadvertently reappear. To prevent this, powerpop stores a failure log per backend which stores failed deletions.

When a backend comes online again, these deletions are first replayed before messages are made available to users again. This prevents messages from magically appearing again.

Chapter 9. Per mailbox configuration items

Each email address within PowerMail has an entry in the Userbase. Any email address offered to PowerMail over the internet is in three categories, 'Unknown', 'Hosted', or 'Forward'. Unkown email addresses are refused immediately - email is not accepted first and bounced later.

Mail for Hosted addresses is stored on any of the `pptalker` backends, and can from there on be retrieved over the POP interface, if the proper password is supplied.

Email is not stored infinitely - a quota can be set which limits the amount of storage any mailbox can occupy for itself. Note that due to the hardlinking nature of PowerMail, it is possible for individual mailboxes to exceed their quota if the messages are shared with other users. This means that a 'message to everybody' will always arrive, even if the user is over quota. See also Section 11.2.

'Forward' addresses are proxied directly to an outgoing mailserver and do not enter PowerMail as such. PowerMail as such does not send out messages, it leaves that job to a outgoing capable mailserver.

9.1. Password schemas

Passwords can be stored in:

- Plaintext ({plain})
- Standard unix crypt(3) ({crypt})
- Standard unix md5 ({md5})

PowerMail 1.0.0 only supported plaintext passwords and these did not need to be prefixed in any way. So, to have a user with password 's3cr3t!', any userbase would contain just 's3cr3t!'. As of version 1.1.0 however, multiple schemas are available.

The schema is indicated by prefixing the password or hash in the database with a marker. For example, the new preferred way to store the password above is '{plain}s3cr3t!'. When using UNIX crypt, it might look like this: '{crypt}/CFF1gJfAFAqM'. When using md5 hashes, a 1 prefix needs to be present, and our database might contain '{md5}1Wh/8PmbX$tLpq3mPsvT5gdVJcVVYXA1' which matches 's3cr3t!'.

Crypts and hashes can be calculated using the **pptool** convenience functions `crypt` and `md5` as described in Section 7.1.1.

Note: Apache htpasswd generates MD5 hashes which are not compatible with PowerMail. PowerMail is compatible with `/etc/shadow` or `/etc/passwd` hashes.

For historical reasons, unprefixed passwords are treated as if they were prefixed by '{plain}'. However, this means that users which previously had passwords that started with a '{' and contained a '}' somewhere are no longer able to login.

Warning

When upgrading from 1.0.0 or earlier, it is highly advised to prefix all plaintext passwords with {plain} in all userbases! Not doing so may cause users with passwords starting with '{' and containg a '}' to not be able to log in!

In the near future, '{sha1}' is also expected to make an appearance.

9.2. Forwarding

To enable forwarding to work, an additional outgoing mailserver is needed. Any one will do. PowerMail 'preforwards' the email so the outgoing mailserver does not need access to the database. To configure a forwarder, set the **smtp-sender-address** and **smtp-sender-port** parameters in `powersmtp.conf`

Make very sure that these settings do not point to another PowerMail configuration as this will lead to bouncing email!

PowerDNS has verified the proper operation of PowerMail with qmail, postfix and sendmail, all fine outgoing mailservers.

9.3. Catch-all addresses

A catch-all address is one that, in the absence of a mailbox with that exact name, receives all mail for the domain in which it resides.

So, if the accounts <`info@example.com`> and <`sales@example.com`> and <`*@example.com`> exist, and mail comes in for <`sales@example.com`>, it goes to that mailbox.

Mail for <`support@example.com`> is handled by the <`*@example.com`> account. This may be a forward or even a mailbox.

To retrieve mail from a catch all mailbox, either use login '*@example.com' or plain 'example.com' or even 'whatever@example.com'.

Chapter 10. The Userbases

PowerMail comes with a number of userbases. Some of those are documented here. The default userbase is currently 'text'. Which userbase to employ should be configured in `power.conf` by setting the **userbase** parameter.

10.1. MySQL Plain

This is purely a mail table with no hooks into the PowerDNS Platform. Use it if email is entirely separate from your DNS products.

This is the userbase that corresponds to the schema as shown in Chapter 4. It is called 'mysqlplain'.

The fields in the `mboxes` table are described below:

id

Not read by PowerMail but may well be used to link mailboxes to other information.

mbox

Full name of this mailbox and also the email address that corresponds to it.

password

Password needed to access this mailbox.

Note: See Section 9.1 for how to encode passwords in plaintext or as a hash.

quotaMB

Quota of this mailbox, in megabytes. 0 stands for unlimited.

isForward

If true, this mailbox is a forward to the address specified in fwdDest. Will only work if an SMTP forwarder has been defined, see Section 9.2.

fwdDest

If this mailbox is forwarded, this should contain a single email address to which email should be forwarded.

10.1.1. Configuration options

This userbase needs to be able to find the MySQL database and connect to it with the proper credentials. For this purpose, the following entries can be set in `power.conf` or, if desired, `powersmtp.conf` and `powerpop.conf`:

mysql-database

: Database name to connect to. Defaults to 'powermail'.

mysql-host

: Location of MySQL installation. Defaults to 127.0.0.1.

mysql-password

: Password to connect with. Defaults to the emtpy string.

mysql-user

: User to connect as. Defaults to the empty string, which means 'current unix username'.

10.2. MySQL PowerDNS

This userbase connects to the database layout as used by the PowerDNS Platform Layer 1. For documentation, see there.

FIXME: write more

10.3. MySQL Simple PowerDNS

This userbase connects to the database layout as used by the PowerDNS 'mysqlbackend'. For users of PowerDNS Platform 0.

FIXME: write more

10.4. Generic PostgreSQL

The PostgreSQL userbase is flexible in that it allows the user to specify a SQL query from the configuration. The default query corresponds to the following schema:

```
create table mboxes (
 id  SERIAL PRIMARY KEY,
 mbox  VARCHAR(255) NOT NULL,
 password  VARCHAR(50) DEFAULT NULL,
 quotaMB  INT DEFAULT NULL,
 isForward       INT DEFAULT NULL,
 fwdDest         VARCHAR(80) DEFAULT NULL
);
CREATE UNIQUE INDEX mbox_index ON mboxes(mbox);
GRANT SELECT ON mboxes TO powermail;
```

The default query is:

```
  select quotaMB,isForward,fwdDest,password from mboxes where mbox='%s'
```

10.4.1. Configuring PostgreSQL connectivity

The following parameters are available to configure the PostgreSQL userbase:

postgresql-database

Database name to connect to. Defaults to 'powermail'.

postgresql-host

Location of PostgreSQL installation. Defaults to 127.0.0.1. Can also be set to a path, a useful setting to try is `/tmp`.

postgresql-password

Password to connect with. Defaults to the emtpy string.

postgresql-user

User to connect as. Defaults to the empty string, which means 'current unix username', often 'root' in the case of PowerMail being launched from bootscripts.

postgresql-query

Query to perform. Can be changed at will but must return data in a specified order, see below.

A typical session setting up PostgreSQL might be:

```
$ createdb powermail
CREATE DATABASE
```

```
$ createuser powermail
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
$ psql powermail
powermail=# create table mboxes (
powermail(#  id SERIAL PRIMARY KEY,
powermail(#  mbox VARCHAR(255) NOT NULL,
powermail(#  password VARCHAR(50) DEFAULT NULL,
powermail(#  quotaMB INT DEFAULT NULL,
powermail(#  isForward       INT DEFAULT NULL,
powermail(#  fwdDest         VARCHAR(80) DEFAULT NULL
powermail(# );
NOTICE:  CREATE TABLE will create implicit sequence 'mboxes_id_seq' for SERIAL column 'mbox
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'mboxes_pkey' for table 'mbo
CREATE
powermail=# CREATE UNIQUE INDEX mbox_index ON mboxes(mbox);
CREATE
powermail=# GRANT SELECT ON mboxes TO powermail;
GRANT
powermail=# insert into mboxes (mbox,password,quotaMB) values ('info@example.com','{plain}s
INSERT 142969 1
```

This gives SELECT rights to a user called 'powermail'. Now add the following to the `power.conf` configuration file:

```
# echo userbase=postgresql > power.conf
# echo postgresql-user=powermail >> power.conf
```

Note: If you are sure that PostgreSQL is running locally but the connection fails with 'could not connect to server: Connection refused', try adding 'postgresql-host=/tmp' to power.conf

Now try launching **powerpop** and **powersmtp**, the user <`info@example.com`> should now exist.

10.4.2. Custom queries & schema description

Any query supplied must return the same values as the default one, and in the same order. The fields are:

mbox

Full name of this mailbox and also the email address that corresponds to it.

password

Password needed to access this mailbox.

Note: See Section 9.1 for how to encode passwords in plaintext or as a hash.

quotaMB

Quota of this mailbox, in megabytes. 0 stands for unlimited.

isForward

If true, this mailbox is a forward to the address specified in fwdDest. Will only work if an SMTP forwarder has been defined, see Section 9.2.

fwdDest

If this mailbox is forwarded, this should contain a single email address to which email should be forwarded.

10.5. Text based

This userbase parses a simple textfile with mailbox information. The default location of this file is `/etc/powermail/mailboxes`. To select, set **userbase=text**.

Sample configuration:

```
/* mailbox with account info@example.com (or info%example.com) */

address "info@example.com" {
  password "{plain}s3cr3t!";  # plaintext!
  quota 250k;
};

address "md5@example.com" {
  password "{md5}$1$shAcLd5E$7VrUnZ46/LkOmychA.Jca0"; # md5 hash of 's3cr3t!'
};
```

```
address "sales@example.com" {
forward "info@example.com";
};
```

Note the need for a semicolon after the closing brace of each completed address statement. Furthermore, each statement within also needs a semicolon.

If no quota is specified, it is infinite. To specify kilobytes, use the suffix 'k', megabytes 'm'.

If this file is changed, it is reread instantly. Any parsing errors are noted in the log and in that case the old configuration is retained in memory. A reparse will be attempted for each new message coming in, or each attempt to check email.

Each new message will cause the file to be reparsed, until it is read succesfully.

Even though this backend is quite simplistic, it parses 50.000 mailboxes within a second on a commodity server. In case a mailbox is present multiple times, behaviour is undefined. The first, last or even middle appearances may be chosen.

10.5.1. Configuration

The location of the textfile is configured with the **text-base** setting.

10.6. RADIUS

Email <pdns@powerdns.com> if you are interested in a RADIUS userbase, in which case we will expedite its development.

10.7. LDAP

Email <pdns@powerdns.com> if you are interested in an LDAP userbase, in which case we will expedite its development.

Chapter 11. In depth technicalities

Each backend is a pplistener daemon. A pplistener stores all messages under one doubly hashed directory, so a typical Maildir might be ’/var/powermail/messages/01/04/info@company.biz/’. Because all mail is stored in a single filesystem, it is possible for a file to be present in many mailboxes, and this is in fact how PowerMail operates.

The operating system will only actually store one (1) copy of a message, no matter how many recipients that message has on that filesystem. It will be deleted when it is removed from the last mailbox that references the message.

This allows for great space savings and, more importantly, for ’straight paper path’ delivery.

11.1. Maildir and queue-less delivery

Each mailbox is more-or-less a standard Qmail Maildir, which means that new messages are written in the ’tmp’ subdirectory and are moved to ’cur’ subdirectory then they are ready.

Because of this, no locking is needed, and messages can be delivered atomically to many recipients, with intermediate failure at worst causing a duplicate delivery.

Because of the use of hardlinks, it is possible to deliver messages straight into the Maildir, without use of a queue in between. This makes mail delivery very fast and robust.

11.2. Quotas

Quotas in PowerMail work somewhat different from other mail solutions. A message with a single recipient is quite standard: all messages on all backends are accounted for, whereby each message is calculated to use 4kb minimum, which is actually in line with the amount of disk space occupied.

If there is room for the new message, it is stored.

However, if a message has multiple recipients, things are a bit different. As PowerMail uses hardlinks to deliver messages space efficiently, a message is accepted if at least *one* of the recipients has enough room in his mailbox.

This has the additional benefit of making sure that ’messages to everybody’ never bounce. Furthermore, users perceive more diskspace than they actually use.

Chapter 12. Best practice setups

An ideal setup consists of at least three servers. This guarantees that a single-machine failure does not hamper service in any way. As there are still two servers available, which together contain all messages, both delivery and retrieval continue. There is no lack of functionality.

A futher failure will cause part of the messages (around 33%) to disappear, but mail retrieval will still be possible. Mail deliveries will cease.

12.1. High redundancy

Six machines in three groups with a redundancy target of 3 means that any two servers can fail before messages might be lost. In other words, 33% of your servers may fail before problems occur.

Mail deliver patterns then look like this:

```
1 2   3 4   5 6
1 x     x     x
2 x     x       x
3 x       x   x
4 x       x     x
5   x   x     x
6   x   x       x
7   x     x   x
8   x     x     x
```

If servers 1, 2 and 3 fail, no mail is lost. In the general case, mail might only be lost if all groups lose a server. Otherwise, at least one undamaged group survives and by our configuration, each group has a copy of all mesages.

The chance for three failures to fall into three groups is 1-((3/3) * (2/3) * (1/3))=1/3. So only in one-third of the cases, there is a chance of all groups losing a message. This only happens if the failures exacly strike the three servers that house your message, for which the chance is (1/2)*(1/2)*(1/2)=1/8.

Summarizing, the compound expected loss of mail in case of 50% failure is (1/3)*(1/8)=1/24, which is around 4%.

There is a 1/3 chance that messages can continue to be stored redundantly, but in that case, 12.5% of already present mail is lost.

There is a 2/3 chance that mail reception is disabled, but not a single message is lost.