PowerDNS manual

PowerDNS BV

pdns@powerdns.com

PowerDNS manual
by

Published v2.1 $Date: 2003/06/28 16:46:01 $

> It is a book about a Spanish guy called Manual. You should read it.
> -- Dilbert

Table of Contents

List of Tables

Chapter 1. The PowerDNS dynamic nameserver

The PowerDNS daemon is a versatile nameserver which supports a large number of backends. These backends can either be plain zonefiles or be more dynamic in nature.

Prime examples of backends include relational databases, but also loadbalancing and failover algorithms.

The company is called PowerDNS BV, the nameserver daemon is called PDNS.

1.1. Function & design of PDNS

PDNS is an authoritative only nameserver. It will answer questions about domains it knows about, but will not go out on the net to resolve queries about other domains. However, it can use a recursing backend to provide that functionality.

When PDNS answers a question, it comes out of the database, and can be trusted as being authoritative. There is no way to pollute the cache or to confuse the daemon.

PDNS has been designed to serve both the needs of small installations by being easy to setup, as well as for serving very large query volumes on large numbers of domains.

Another prime goal is security. By the use of language features, the PDNS source code is very small (in the order of 10.000 lines) which makes auditing easy. In the same way, library features have been used to mitigate the risks of buffer overflows.

Finally, PDNS is able to give a lot of statistics on its operation which is both helpful in determining the scalability of an installation as well as for spotting problems.

1.2. About this document

If you are reading this document from disk, you may want to check http://doc.powerdns.com for updates. The PDF version is available on http://doc.powerdns.com/pdf, a text file is on http://doc.powerdns.com/txt/ (http://doc.powerdns.com/txt).

1.3. Release notes

Before proceeding, it is advised to check the release notes for your PDNS version, as specified in the name of the distribution file.

1.3.1. Version 2.9.10

Small bugfixes, LDAP update. Released 3rd of July 2003. Apologies for the long delay, real life keeps interfering.

> **Warning**
>
> Do not use or try to use 2.9.9, it was a botched release!

> **Warning**
>
> There has been a change in behaviour whereby **disable-axfr** does what it means now! From now on, setting **allow-axfr-ips** automatically disables AXFR from unmentioned subnets.

- 2.9.8 was prone to crash on adding additional records. Thanks to excellent debugging by PowerDNS users worldwide, the bug was found quickly and is in fact present in all earlier PowerDNS releases, but for some reason doesn't cause crashes there.
- Notifications now jump in front of the queue of domains that need to be checked for changes, giving much greater perceived performance. This is needed if you have tens of thousands of slave domains and your master server is on a high latency link. Thanks to Mark Jeftovic of EasyDNS for suggesting this change and testing it on their platform.
- Dean Mills reported that PowerDNS does confusing logging about changing GIDs and UIDs, fixed. Cosmetic only.
- pdns_recursor may have logged empty lines for some users, fixed. Solution suggested by Norbert Sendetzky.
- LDAP: DNS TTLs were random values (Norbert Sendetzky, Stefan Pfetzing). New **ldap-default-ttl** option.
- LDAP: Now works with OpenLDAP 2.1 (Norbert Sendetzky)
- LDAP: error handling for invalid MX records implemented (Norbert Sendetzky)
- LDAP: better exception handling (Norbert Sendetzky)
- LDAP: code cleanup of lookup() (Norbert Sendetzky)
- LDAP: added support for scoped searches (Norbert Sendetzky)

1.3.2. Version 2.9.8

Queen's day release! 30th of April 2003.

Added support for AIX, fixed negative SOA caching. Some other cleanups. Not a major release but enough reasons to upgrade.

Bugs fixed:

- Recursor had problems expiring negatively cached entries, which wasted memory and also led to the continued non-existence of hosts that since had come into existence.
- The Generic SQL backends did not lowercase the names of records, which led to new records not being found by case sensitive databases (notably PostgreSQL). Found by Volker Goetz.
- NS queries for zones for which we did not carry authority, but only had delegation information, had their NS records in the wrong section. Minor detail, but a standards violation on etheless. Spotted by Stephane Bortzmeyer.

Improvements:

- Removed crypt.h dependency from powerldap.hh, which was a problem on some platforms (Richard Arends)
- PowerDNS can't parse so called binary labels which we now detect and ignore, after printing a warning.
- Specifying allow-axfr-ips now automatically disables AXFR for all non-mentioned addresses.
- A Solaris ready init.d script is now part of the tar.gz (contributed, but I lost by whom).
- Added some fixes to PowerDNS can work on AIX (spotted by Markus Heimhilcher).
- Norbert Sendetzky contributed `zone2ldap`.
- Everybody's favorite compiler warning from `zone2sql.cc` was removed!
- Recursor now listens on TCP!

1.3.3. Version 2.9.7

Released on 2003-03-20.

This is a sweeping release in the sense of cleanup. There are some new features but mostly a lot of cleanup going on. Hiding inside is the `bind2backend`, the next generation of the bind backend. A work in progress. Those of you with overlapping zones, as mentioned in the changelog of 2.9.6, are invited to check it out by replacing **launch=bind** by **launch=bind2** and renaming all **bind-** parameters to **bind2-**.

Be aware that if you run with many small zones, this backend is faster, but if you run with a few large ones, it is slower. This will improve.

Features:

- Mark Bergsma contributed **query-local-address** which allows the operator to select which source address to use. This is useful on servers with multiple source addresses and the operating system selecting an unintended one, leading to remotes denying access.
- PowerDNS can now perform AAAA additional processing optionally, turned on by setting **do-ipv6-additional-processing**. Thanks to Stephane Bortzmeyer for pointing out the need.
- Bind2backend, which is almost in compliance with the new IETF AXFR-clarify (some would say 'redefinition') draft.

  This backend is not ready for primetime but you may want to try it if you currently have overlapping zones and note problems. An overlapping zone would be having "ipv6.powerdns.com" and "powerdns.com" zones on one server.

Improvements:

- Zone2sql would happily try to read from a directory and not give a useful error about this.
- PowerDNS now reports the case where it can't figure out any IP address of slave nameservers for a zone
- Removed **receiver-threads** setting which was experimental and in fact only made things worse.
- LDAP backend updates from its author Norbert Sendetzky. Reverse lookups should work now too.
- An error message about unparseable packets did not include the originating IP address (fixed by Mark Bergsma)
- PowerDNS can now be started via path resolution while running with a guardian. Suggested by Maurice Nonnekes.
- `pdns_recursor` moved to `sbin` (reported by Norbert Sendetzky)
- Retuned some logger errorlevels, a lot of master/slave chatter was logged as 'Error'. Reported by Willem de Groot.

Bugs fixed:

- `zone2sql` did not remove trailing dots in SOA records.
- ldapbackend did not include `utility.hh` which caused compilation problems on Solaris (reported by Remco Post)
- `pdns_control` could leave behind remnants in case PowerDNS was not running (reported by dG)

- Incoming AXFR did not work on Solaris and other big-endian systems (Willem de Groot helped debugging this long standing problem).
- Recursor could crash on convoluted CNAME loops. Thanks to Dan Faerch for delivering coredumps.
- Silly 'wuh' debugging output in zone2sql and bindbackend removed (spotted by Ivo van der Wijk)
- Recursor neglected to differentiate between negative cache of NXDOMAIN and NOERROR, leading to problems with IPv6 enabled Windows clients. Thanks to Stuart Walsh for reporting this and testing the fix.
- PowerDNS set the 'aa' bit on serving NS records in a zone for which it was authoritative. Most implementations drop the 'aa' bit in this case and Stephane Bortzmeyer informed us of this. PowerDNS now also drops the 'aa' bit in this case.
- The webserver tended to fail after prolonged operation on FreeBSD, this was due to an uninitialised timeout, other platforms were lucky. Thanks to G.P. de Boer for helping debug this.
- getAnswers() in dnspacket.cc could be forced to read bytes beyond the end of the packet, leading to crashes in the PowerDNS recursor. This is an ongoing project that needs more work. Reported by Dan Faerch, with a coredump proving the problem.

1.3.4. Version 2.9.6

Two new backends - Generic ODBC (windows only) and LDAP. Furthermore, a few important bugs have been fixed which may have hampered sites seeing a lot of outgoing zonetransfers. Additionally, the pdns recursor now has 'query throttling' which is pretty cool. In short this makes sure that PowerDNS does not send out heaps of queries if a nameserver is unable to provide an answer. Many operators of authoritative setups are all too aware of recursing nameservers that hammer them for zones they don't have, PowerDNS won't do that anymore now, no matter what clients request of it.

Warning

There is an unresolved issue with the BIND backend and 'overlapping' slave zones. So if you have 'example.com' and also have a separate slave zone called 'external.example.com', things may go wrong badly. Thanks to Christian Laursen for working with us a lot in finding this issue. We hope to resolve it soon.

- BIND Backend now honours notifies, code to support this was accidentally left out. Thanks to Christian Laursen for noticing this.
- Massive speedup for those of you using the slightly deprecated MBOXFW records. Thanks to Jorn of ISP Services (http://www.ISP-Services.nl) for helping and testing this improvement.

- $GENERATE had an off-by-one bug where it would omit the last record to be generated (Christian Laursen)
- Simultaneous AXFRs may have been problematic on some backends. Thanks to Jorn of ISP-Services again for helping us resolve this issue.
- Added LDAP backend by Norbert Sendetzky, see Section A.11.
- Added Generic ODBC backend for Windows by Michel Stol.
- Simplified 'out of zone data' detection in incoming AXFR support, hopefully removing a case sensitivity bug there. Thanks again to Christian Laursen for reporting this issue.
- $include in-zonefile was broken under some circumstances, losing the last character of a filename. Thanks to Joris Vandalon for noticing this.
- The zoneparser was more case-sensitive than BIND, refusing to accept 'in' as well as 'IN'. Thanks to Joris Vandalon for noticing this.

1.3.5. Version 2.9.5

Released on 2002-02-03.

This version is almost entirely about recursion with major changes to both the pdns recursor, which is renamed to '`pdns_recursor`' and to the main PowerDNS binary to make it interact better with the recursing component.

Sadly, due to technical reasons (http://sources.redhat.com/ml/libc-alpha/2003-01/msg00245.html), compiling the pdns recursor and pdns authoritative nameserver into one binary is not immediately possible. During the release of 2.9.4 we stated that the recursing nameserver would be integrated in the next release - this won't happen now.

However, this turns out to not be that bad at all. The recursor can now be restarted without having to restart the rest of the nameserver, for example. Cooperation between the both halves of PDNS is also almost seamless. As a result, 'non-lazy recursion' has been dropped. See Chapter 11 for more details.

Furthermore, the recursor only works on Linux, Windows and Solaris (not entirely). FreeBSD does not support the required functions. If you know any important FreeBSD people, plea with them to support set/get/swapcontext! Alternatively, FreeBSD coders could read the solution presented here in figure 5 (http://www.eng.uwaterloo.ca/~ejones/software/threading.html).

The 'Contributor of the Month' award goes to Mark Bergsma who has responded to our plea for help with the label compressor and contributed a wonderfully simple and right fix that allows PDNS to compress just as well as Other namerervers out there. An honorary mention goes to Ueli Heuer who, despite having no C++ experience, submitted an excellent SRV record implementation.

Excellent work was also performed by Michel Stol, the Windows guy, in fixing all our non-portable stuff again. Christof Meerwald has also done wonderful work in porting MTasker to Windows, which was then used by Michel to get the recursor functioning on Windows.

Other changes:

- dnspacket.cc was cleaned up by factoring out common operations
- Heaps of work on the recursing nameserver. Has now achieved *days* of uptime!
- Recursor renamed from syncres to `pdns_recursor`
- PowerDNS can now serve records it does not know about. To benefit from this slightly undocumented feature, add 1024 to the numerical type of a record and include the record in binary form in your database. Used internally by the recursing nameserver but you can use it too.
- PowerDNS now knows about SIG and KEY records *names*. It does not support them yet but can at least report so now.
- HINFO records can now be transferred from a master to PowerDNS (thanks to Ueli Heuer for noticing it didn't work).
- Yet more UltraSPARC alignment issues fixed (Chris Andrews).
- Dropped non-lazy recursion, nobody was using it. Lazy recursion became even more lazy after Dan Bernstein pointed out that additional processing is not vital, so PowerDNS does its best to do additional processing on recursive queries, but does not scream murder if it does not succeed. Due to caching, the next identical query will be successfully additionally processed.
- Label compression was improved so we can now fit all . records in 436 bytes, this used to be 460! (Code & formal proof of correctness by Mark Bergsma).
- SRV support (incoming and outgoing), submitted by Ueli Heuer.
- Generic backends do not support SOA serial autocalculation, it appears. Could lead to random SOA serials in case of a serial of 0 in the database. Fixed so that 0 stays zero in that case. Don't set the SOA serial to 0 when using Generic MySQL or Generic PostgreSQL!
- J root-server address was updated to its new location.
- SIGUSR1 now forces the recursor to print out statistics to the log.
- Meaning of recursor logging was changed a bit - a cache hit is now a question that was answered with 0 outgoing packets needed. Used to be a weighted average of internal cache hits.
- MySQL compilation did not include -lz which causes problems on some platforms. Thanks to James H. Cloos Jr for reporting this.
- After a suggestion by Daniel Meyer and Florus Both, the built in webserver now reports the configuration name when multiple PowerDNS instances are active.
- Brad Knowles noticed that zone2sql had problems with the root.zone, fixed. This also closes some other zone2sql annoyances with converting single zones.

1.3.6. Version 2.9.4

Yet another grand release. Big news is the addition of a recursing nameserver which has sprung into existence over the past week. It is in use on several computers already but it is not ready for prime time. Complete integration with PowerDNS is expected around 2.9.5, for now the recursor is a separate program.

In preliminary tests, the recursor appears to be four times faster than BIND 9 on a naive benchmark starting from a cold cache. BIND 9 managed to get through to some slower nameservers however, which were given up on by PowerDNS. We will continue to tune the recursor. See Chapter 12 for further details.

The BIND Backend has also been tested (see the **bind-domain-status** item below) rather heavily by several parties. After some discussion online, one of the BIND authors ventured that the newsgroup comp.protocols.dns.bind may now in fact be an appropriate venue for discussing PowerDNS. Since this discussion, traffic to the PowerDNS pages has increased sixfold and shows no signs of slowing down.

From this, it is apparent that far more people are interested in PowerDNS than yet know about it. So spread the word!

In other news, we now have a security page at Section 1.4. Furthermore, Maurice Nonnekes contributed an OpenBSD port! See his page (http://www.codeninja.nl/openbsd/powerdns/) for more details!

New features and improvements:

- All SQL queries in the generic backends are now available for configuration. (Martin Klebermass/bert hubert). See Section A.5.
- A recursing nameserver! See Chapter 12.
- An incoming AXFR now only starts a backend zone replacement transaction after the first record arrived successfully, thus making sure no work is done when a remote nameserver is unable/unwilling to AXFR a zone to us.
- Zoneparser error messages were improved slightly (thanks to Stef van Dessel for spotting this shortcoming)
- XS4ALL's Erik Bos checked how PowerDNS reacted to a BIND installation with almost 60.000 domains, some of which with >100.000 records, and he discovered the pdns_control **bind-domain-status** command became very slow with larger numbers of domains. Fixed, 60.000 domains are now listed in under one second.
- If a remote nameserver disconnects during an incoming AXFR, the update is now rolled back, unless the AXFR was properly terminated.
- The migration chapter mentioned the use of deprecated backends.

A tremendous number of bugs were discovered and fixed:

- Zone parser would only accept $include and not $INCLUDE
- Zone parser had problems with $lines with comments on the end
- Wildcard ANY queries were broken (thanks Colemarcus for spotting this)
- A connection failure with the Generic backends would lead to a powerdns reload (cast of many)
- Generic backends had some semantic problems with slave support. Symptoms were oft-repeated notifications and transfers (thanks to Mark Bergsma for helping resolve this).
- Solaris version compiles again. Thanks to Mohamed Lrhazi for reporting that it didn't.
- Some UltraSPARC alignment fixes. Thanks to Mohamed Lrhazi for being helpful in spotting these. One problem is still outstanding, Mohamed sent a core dump that tells us where the problem is. Expect the fix to be in 2.9.5. Volunteers can grep the source for 'UltraSPARC' to find where the problem is.
- Our support of IPv6 on FreeBSD had phase of moon dependent bugs, fixed by Peter van Dijk.
- Some crashes of and by pdns_control were fixed, thanks to Mark Bergsma for helping resolve these.
- Outgoing AXFR in pdns installations with multiple loaded backends was broken (thanks to Stuart Walsh for reporting this).
- A failed BIND Backend incoming AXFR would block the zone until it succeeded again.
- Generic PostgreSQL backend wouldn't compile with newer libpq++, fixed by Julien Lemoine/SpeedBlue.
- Potential bug (not observed) when listening on multiple interfaces fixed.
- Some typos in manpages fixed (reported by Marco Davids).

1.3.7. Version 2.9.3a

Note: 2.9.3a is identical to 2.9.3 except that zone2sql does work

Broad range of huge improvements. We now have an all-static .rpm and .deb for Linux users and a a link to an OpenBSD port. Major news is that work on the Bind backend has progressed to the point that we've just retired our last Bind server and replaced it with PowerDNS in Bind mode! This server is operating a number of master and slave setups so it should stress the Bind backend somewhat.

This version is rapidly approaching the point where it is a better-Bind-than-Bind and nearly a drop-in replacement for authoritative setups. PowerDNS is now equipped with a powerful master/slave apparatus that offers a lot of insight and control to the user, even when operating from Bind zonefiles and a Bind configuration. Observe.

After the SOA of ds9a.nl was raised:

```
pdns[17495]: All slave domains are fresh
pdns[17495]: 1 domain for which we are master needs notifications
pdns[17495]: Queued notification of domain 'ds9a.nl' to 195.193.163.3
pdns[17495]: Queued notification of domain 'ds9a.nl' to 213.156.2.1
pdns[17520]: AXFR of domain 'ds9a.nl' initiated by 195.193.163.3
pdns[17520]: AXFR of domain 'ds9a.nl' to 195.193.163.3 finished
pdns[17521]: AXFR of domain 'ds9a.nl' initiated by 213.156.2.1
pdns[17521]: AXFR of domain 'ds9a.nl' to 213.156.2.1 finished
pdns[17495]: Removed from notification list: 'ds9a.nl' to 195.193.163.3 (was acknowledged)
pdns[17495]: Removed from notification list: 'ds9a.nl' to 213.156.2.1 (was acknowledged)
pdns[17495]: No master domains need notifications
```

If however our slaves would ignore us, as some are prone to do, we can send some additional notifications:

```
$ sudo pdns_control notify ds9a.nl
Added to queue
pdns[17492]: Notification request for domain 'ds9a.nl' received
pdns[17492]: Queued notification of domain 'ds9a.nl' to 195.193.163.3
pdns[17492]: Queued notification of domain 'ds9a.nl' to 213.156.2.1
pdns[17495]: Removed from notification list: 'ds9a.nl' to 195.193.163.3 (was acknowledged)
pdns[17495]: Removed from notification list: 'ds9a.nl' to 213.156.2.1 (was acknowledged)
```

Conversely, if PowerDNS needs to be reminded to retrieve a zone from a master, a command is provided:

```
$ sudo pdns_control retrieve forfun.net
Added retrieval request for 'forfun.net' from master 212.187.98.67
pdns[17495]: AXFR started for 'forfun.net', transaction started
pdns[17495]: Zone 'forfun.net' (/var/cache/bind/forfun.net) reloaded
pdns[17495]: AXFR done for 'forfun.net', zone committed
```

Also, you can force PowerDNS to reload a zone from disk immediately with **pdns_control bind-reload-now**. All this happens 'live', per your instructions. Without instructions, the right things also happen, but the operator is in charge.

For more about all this coolness, see Section B.1.1 and Section A.8.2.

Warning

Again some changes in compilation instructions. The hybrid pgmysql backend has been split up into 'gmysql' and 'gpgsql', sharing a common base within the PowerDNS server itself. This means that you can no longer compile **--with-modules="pgmysql" --enable-mysql --enable-pgsql** but that you should now use: **--with-modules="gmysql gpgsql"**. The old launch-names remain available.

If you launch the Generic PgSQL backend as gpgsql2, all parameters will have gpsql2 as a prefix, for example **pgsql2-dbname**. If launched as gpsql, the regular names are in effect.

Warning

The pdns_control protocol was changed which means that older pdns_controls cannot talk to 2.9.3. The other way around is broken too. This may lead to problems with automatic upgrade scripts, so pay attention if your daemon is truly restarted.

Also make sure no old pdns_control command is around to confuse things.

Improvements:

- Bind backend can now deal with missing files and try to find them later.
- Bind backend is now explicitly master capable and triggers the sending of notifications.
- General robustness improvements in Bind backend - many errors are now non-fatal.
- Accessability, Serviceability. New **pdns_server** commands like **bind-list-rejects** (lists zones that could not be loaded, and the reason why), **bind-reload-now** (reload a zone from disk NOW), **rediscover** (reread named.conf NOW). More is coming up.
- Added support for retrieving RP (Responsible Person) records from remote masters. Serving them was already possible.
- Added support for LOC records, which encode the geographical location of a host, both serving and retrieving (thanks to Marco Davids using them on our last Bind server, forcing us to implement this silly record).
- Configuration file parser now strips leading spaces too, allowing "chroot= /tmp" to work, as well as "chroot=/tmp" (Thanks to Hub Dohmen for reporting this for months on end).

- Added **bind-domain-status** command that shows the status of all domains (when/if they were parsed, any errors encountered while parsing them).
- Added **bind-reload-now** command that tries to reload a zone from disk NOW, and reports back errors to the operator immediatly.
- Added **retrieve** command that queues a request to retrieve a zone from its master.
- Zones retrieved from masters are now stored way smaller on disk because the domain is stripped from records, which is derived from the configuration file. Retrieved zones are now prefixed with some information on where they came from.

Changes:

- gpgsql and gmysql backends split out of the hybrid pgmysqlbackend. This again changed compilation instructions!
- **pdns_control** now uses the rarely seen SOCK_STREAM Unix Domain socket variety so it can transport large amounts of text, which is needed for the **bind-domain-status** command, for which see Section A.8.2. This breaks compatability with older pdns_control and pdns_server binaries!
- Bind backend now ignores 'hint' and 'forward' and other unsupported zone types.
- AXFRs are now logged more heavily by default. An AXFR is a heavy operation anyhow, some more logging does not further increase the load materially. Does help in clearing up what slaves are doing.
- A lot of master/slave chatter has been silenced, making output more relevant. No more repetitive 'No master domains need notifications' etc, only changes are reported now.

Bugfixes:

- Windows version did not compile without minor changes.
- Confusing error reporting on Windows 98 (which does not support PowerDNS) fixed
- Potential crashes with shortened packets addressed. An upgrade is advised!
- **notify** (which was already there, just badly documented) no longer prints out debugging garbage.
- pgmysql backend had problems launching when not compiled in but available as a module. Workaround for 2.9.2 is 'load-modules=pgmysql', but even then gpgsql would not work! gmysql would then, however. These modules are now split out, removing such issues.

1.3.8. Version 2.9.2

Bugfixes galore. Solaris porting created some issues on all platforms. Great news is that PowerDNS is now in Debian 'sid' (unstable). The 2.9.1 packages in there currently aren't very good but the 2.9.2 ones will be. Many thanks to Wichert Akkerman, our 'downstream' for making this possible.

> **Warning**
>
> The Generic MySQL backend, part of the Generic MySQL & PostgreSQL backend, is now the DEFAULT! The previous default, the 'mysql' backend (note the lack of 'g') is now DEPRECATED. This was the source of much confusion. The 'mysql' backend does not support MASTER or SLAVE operation. The Generic backends do.
>
> To get back the mysql backend, add --with-modules="mysql" or --with-dynmodules="mysql" if you prefer to load your modules at runtime.

Bugs fixed:

- Silly debugging output removed from the webserver (found by Paul Wouters)
- SEVERE: due to Solaris portability fixes, qtypes<127 were broken. These include NAPTR, ANY and AXFR. The upshot is that powerdns wasn't performing outgoing AXFRs nor ANY queries. These were the 'question for type -1' warnings in the log
- incoming AXFR could theoretically miss some trailing records (not observed, but could happen)
- incoming AXFR did not support TXT records (spotted by Paul Wouters)
- with some remotes, an incoming AXFR would not terminate until a timeout occured (observed by Paul Wouters)
- Documentation bug, pgmysql != mypgsql

Documentation:

- Documented the 'random backend', see Section A.3.
- Wichert Akkerman contributed three manpages.
- Building PowerDNS on Unix is now documented somewhat more, see Section D.1.

Features:

- pdns init.d script is now +x by default
- OpenBSD is on its way of becoming a supported platform! As of 2.9.2, PowerDNS compiles on OpenBSD but swiftly crashes. Help is welcome.
- ODBC backend (for Windows only) was missing from the distribution, now added.
- xdb backend added - see Section A.10. Designed for use by root-server operators.

- Dynamic modules are back which is good news for distributors who want to make a pdns packages that does not depend one every database under the sun.

1.3.9. Version 2.9.1

Thanks to the great enthusiasm from around the world, powerdns is now available for Solaris and FreeBSD users again! Furthermore, the Windows build is back. We are very grateful for the help of:

- Michel Stol
- Wichert Akkerman
- Edvard Tuinder
- Koos van den Hout
- Niels Bakker
- Erik Bos
- Alex Bleker
- steven stillaway
- Roel van der Made
- Steven Van Steen

We are happy to have been able to work with the open source community to improve PowerDNS!

Changes:

- The monitor command **set** no longer allows the changing of non-existant variables.
- IBM Universal Database DB2 backend now included in source distribution (untested!)
- Oracle backend now included in source distribution (sligthly tested!)
- configure script now searches for postgresql and mysql includes
- Bind parser now no longer dies on records with a ’ in them (Erik Bos)
- The pipebackend was accidentally left out of 2.9
- FreeBSD fixes (with help from Erik Bos, Alex Bleeker, Niels Bakker)
- Heap of Solaris work (with help from Edvard Tuinder, Stefan Van Steen, Koos van den Hout, Roel van der Made and especially Mark Bakker). Now compiles in 2.7 and 2.8, haven’t tried 2.9. May be a bit dysfunctional on 2.7 though - it won’t do IPv6 and it won’t serve AAAA. Patches welcome!
- Windows 32 build is back! Michel Stol updated his earlier work to the current version.
- S/Linux (Linux on Sparc) build works now (with help from steven stillaway).

- Silly debugging message ('sd.ttl from cache') removed
- .debs are back, hopefully in 'sid' soon! (Wichert Akkerman)
- Removal of bzero and other less portable constructs. Discovered that recent Linux glibc's need -D_GNU_SOURCE (Wichert Akkerman).

1.3.10. Version 2.9

Open source release. Do not deploy unless you know what you are doing. Stability is expected to return with 2.9.1, as are the binary builds.

- License changed to the GNU General Public License version 2.
- Cleanups by Erik Bos @ xs4all.
- Build improvements by Wichert Akkerman
- Lots of work on the build system, entirely revamped. By PowerDNS.

1.3.11. Version 2.8

From this release onwards, we'll concentrate on stabilising for the 3.0 release. So if you have any must-have features, let us know soonest. The 2.8 release fixes a bunch of small stability issues and add two new features. In the spirit of the move to stability, this release has already been running 24 hours on our servers before release.

- pipe backend gains the ability to restricts its invocation to a limited number of requests. This allows a very busy nameserver to still serve packets from a slow perl backend.
- pipe backend now honors query-logging, which also documents which queries were blocked by the regex.
- pipe backend now has its own backend chapter.
- An incoming AXFR timeout at the wrong moment had the ability to crash the binary, forcing a reload. Thanks to our bug spotting champions Mike Benoit and Simon Kirby of NetNation for reporting this.

1.3.12. Version 2.7 and 2.7.1

This version fixes some very long standing issues and adds a few new features. If you are still running 2.6, upgrade yesterday. If you were running 2.6.1, an upgrade is still strongly advised.

Features:

- The controlsocket is now readable and writable by the 'setgid' user. This allows for non-root access to PDNS which is nice for mrtg or cricket graphs.
- MySQL backend (the non-generic one) gains the ability to read from a different table using the **mysql-table** setting.
- pipe backend now has a configurable timeout using the **pipe-timeout** setting. Thanks fo Steve Bromwich for pointing out the need for this.
- Experimental backtraces. If PowerDNS crashes, it will log a lot of numbers and sometimes more to the syslog. If you see these, please report them to us. Only available under Linux.

Bugs:

- 2.7 briefly broke the mysql backend, so don't use it if you use that. 2.7.1 fixes this.
- SOA records could sometimes have the wrong TTL. Thanks to Jonas Daugaard for reporting this.
- An ANY query might lead to duplicate SOA records being returned under exceptional circumstances. Thanks to Jonas Daugaard for reporting this.
- Underlying the above bug, packet compression could sometimes suddenly be turned off, leading to overly large responses and non-removal of duplicate records.
- The **allow-axfr-ips** setting did not accept IP ranges (1.2.3.0/24) which the documentation claimed it did (thanks to Florus Both of Ascio technologies for being sufficiently persistent in reporting this).
- Killed backends were not being respawned, leading to suboptimal behaviour on intermittent database errors. Thanks to Steve Bromwich for reporting this.
- Corrupt packets during an incoming AXFR when acting as a slave would cause a PowerDNS reload instead of just failing that AXFR. Thanks to Mike Benoit and Simon Kirby of NetNation for reporting this.
- Label compression in incoming AXFR had problems with large offsets, causing the above mentioned errors. Thanks to Mike Benoit and Simon Kirby of NetNation for reporting this.

1.3.13. Version 2.6.1

Quick fix release for a big cache problem.

1.3.14. Version 2.6

Performance release. A lot of work has been done to raise PDNS performance to staggering levels in order to take part in benchmarketing efforts. Together with our as yet unnamed partner, PDNS has been benchmarked at 60.000 mostly cached queries/second on off the shelf PC hardware. Uncached performance was 17.000 uncached DNS queries/second on the .ORG domain.

Performance has been increased by both making PDNS itself quicker but also by lowering the number of backend queries typically needed. Operators will typically see PDNS taking less CPU and the backend seeing less load.

Furthermore, some real bugs were fixed. A couple of undocumented performance switches may appear in --help output but you are advised to stay away from these.

Developers: this version needs the pdns-2.5.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

Performance:

- A big error in latency calculations - cached packets were weighed 50 times less, leading to inflated latency reporting. Latency calculations are now correct and way lower - often in the microseconds range.
- It is now possible to run with 0 second cache TTLs. This used to cause very frequent cache cleanups, leading to performance degradation.
- Many tiny performance improvements, removing duplicate cache key calculations, etc. The cache itself has also been reworked to be more efficient.
- First 'CNAME' backend query replaced by an 'ANY' query, which most of the time returns the actual record, preventing the need for a separate CNAME lookup, halving query load.
- Much of the same for same-level-NS records on queries needing delegation.

Bugs fixed:

- Incidentally, the cache count would show 'unknown' packets, which was harmless but confusing. Thanks to Mike and Simon of NetNation for reporting this.
- SOA hostmaster with a . in the local-part would be cached wrongly, leading to a stray backslash in case of multiple successively SOA queries. Thanks to Ascio Techologies for spotting this bug.
- zone2sql did not parse Verisign zonefiles correctly as these contained a $TTL statement in mid-record.
- Sometimes packets would not be accounted, leading to 'udp-queries' and 'udp-answers' divergence.

Features:

- 'cricket' command added to init.d scripts that provides unadorned output for parsing by 'Cricket'.

1.3.15. Version 2.5.1

Brown paper bag (http://www.tuxedo.org/~esr/jargon/html/entry/brown-paper-bag-bug.html) release fixing a huge memory leak in the new Query Cache.

Developers: this version needs the new pdns-2.5.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

And some small changes:

- Added support for RFC2038 compliant negative-answer caching. This allows remotes to cache the fact that a domain does not exist and will not exist for a while. Thanks to Chris Thompson for pointing out how tiny our minds are (http://ops.ietf.org/lists/namedroppers/namedroppers.2002/msg01697.html). This feature may cause a noticeable reduction in query load.
- Small speedup to non-packet-cached queries, incidentally fixing the huge memory leak.
- **pdns_control ccounts** command outputs statistics on what is in the cache, which is useful to help optimize your caching strategy.

1.3.16. Version 2.5

An important release which has seen quite a lot of trial and error testing. As a result, PDNS can now run with a huge cache and concurrent invalidations. This is useful when running of a slower database or under high traffic load with a fast database.

Furthermore, the gpgsql2 backend has been validated for use and will soon supplant the gpgsql backend entirely. This also bodes well for the gmysql backend which is the same code.

Also, a large amount of issues biting large scale slave operators were addressed. Most of these issues would only show up after prolonged uptime.

New features:

- Query cache. The old Packet Cache only cached entire questions and their answers. This is very CPU efficient but does not lead to maximum hitrate. Two packets both needing to resolve smtp.you.com internally would not benefit from any caching. Furthermore, many different DNS queries lead to the same backend queries, like 'SOA for .COM?'.

  PDNS now also caches backend queries, but only those having no answer (the majority) and those having one answer (almost the rest).

In tests, these additional caches appear to halve the database backend load numerically and perhaps even more in terms of CPU load. Often, queries with no answer are more expensive than those having one.

The default **ttl**s for the query-cache and negquery-cache are set to safe values (20 and 60 seconds respectively), you should be seeing an improvement in behaviour without sacrificing a lot in terms of quick updates.

The webserver also displays the efficiency of the new Query Cache.

The old Packet Cache is still there (and useful) but see Chapter 9 for more details.

- There is now the ability to shut off some logging at a very early stage. High performance sites doing thousands of queries/second may in fact spend most of their CPU time on attempting to write out logging, even though it is ignored by syslog. The new flag **log-dns-details**, on by default, allows the operator to kill most informative-only logging before it takes any cpu.
- Flags which can be switched 'on' and 'off' can now also be set to 'off' instead of only to 'no' to turn them off.

Enhancements:

- Packet Cache is now case insensitive, leading to a higher hitrate because identical queries only differing in case now both match. Care is taken to restore the proper case in the answer sent out.
- Packet Cache stores packets more efficiently now, savings are estimated at 50%.
- The Packet Cache is now asynchronous which means that PDNS continues to answer questions while the cache is busy being purged or queried. Incidentally this will mean a cache miss where previously the question would wait until the cache became available again.

The upshot of this is that operators can call **pdns_control purge** as often as desired without fearing performance loss. Especially the full, non-specific, purge was speeded up tremendously.

This optimization is of little merit for small sites but is very important when running with a large packetcache, such as when using recursion under high load.

- AXFR log messages now all contain the word 'AXFR' to ease grepping.
- Linux static version now compiled with gcc 3.2 which is known to output better and faster code than the previously used 3.0.4.

Bugs fixed:

- Packetcache would sometimes send packets back with slightly modified flags if these differed from the flags of the cached copy.
- Resolver code did bad things with filedescriptors leading to fd exhaustion after prolonged uptimes and many slave SOA currency checks.
- Resolver code failed to properly log some errors, leading to operator uncertainty regarding to AXFR problems with remote masters.
- After prolonged uptime, slave code would try to use privileged ports for originating queries, leading to bad replication efficiency.
- Masters sending back answers in differing case from questions would lead to bogus 'Master tried to sneak in out-of-zone data' errors and failing AXFRs.

1.3.17. Version 2.4

Developers: this version is compatible with the pdns-2.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

This version fixes some stability issues with malformed or malcrafted packets. An upgrade is advised. Furthermore, there are interesting new features.

New features:

- Recursive queries are now also cached, but in a separate namespace so non-recursive queries don't get recursed answers and vice versa. This should mean way lower database load for sites running with the current default lazy-recursion. Up to now, each and every recursive query would lead to a large amount of SQL queries.

  To prevent the packetcache from becoming huge, a separate **recursive-cache-ttl** can be specified.

- The ability to change parameters at runtime was added. Currently, only the new **query-logging** flag can be changed.
- Added **query-logging** flag which hints a backend that it should output a textual representation of queries it receives. Currently only gmysql and gpgsql2 honor this flag.
- Gmysql backend can now also talk to PgSQL, leading to less code. Currently, the old postgresql driver ('gpgsql') is still the default, the new driver is available as 'gpgsql2' and has the benefit that it does query logging. In the future, gpgsql2 will become the default gpgsql driver.
- DNS recursing proxy is now more verbose in logging odd events which may be caused by buggy recursing backends.
- Webserver now displays peak queries/second 1 minute average.

Bugs fixed:

- Failure to connect to database in master/slave communicator thread could lead to an unclean reload, fixed.

Documentation: added details for **strict-rfc-axfrs**. This feature can be used if very old clients need to be able to do zone transfers with PDNS. Very slow.

1.3.18. Version 2.3

Developers: this version is compatible with the pdns-2.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

This release adds the Generic MySQL backend which allows full master/slave semantics with MySQL and InnoDB tables (or other tables that support transactions). See Section A.5.

Other new features:

- Improved error messages in master/slave communicator will help down track problems.
- **slave-cycle-interval** setting added. Very large sites with thousands of slave domains may need to raise this value above the default of 60. Every cycle, domains in undeterminate state are checked for their condition. Depending on the health of the masters, this may entail many SOA queries or attempted AXFRs.

Bugs fixed:

- 'pdns_control purge **`domain`**' and 'pdns_control purge **`domain$`**' were broken in version 2.2 and did not in fact purge the cache. There is a slight risk that domain-specific purge commands could force a reload in previous version. Thanks to Mike Benoit of NetNation for discovering this.
- Master/slave communicator thread got confused in case of delayed answers from slow masters. While not causing harm, this caused inefficient behaviour when testing large amounts of slave domains because additional 'cycles' had to pass before all domains would have their status ascertained.
- Backends implementing special SOA semantics (currently only the undocumented 'pdns express backend', or homegrown backends) would under some circumstances not answer the SOA record in

case of an ANY query. This should put an end to the last DENIC problems. Thanks to DENIC for helping us find the problem.

1.3.19. Version 2.2

Developers: this version is compatible with the pdns-2.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

Again a big release. PowerDNS is seeing some larger deployments in more demanding environments and these are helping shake out remaining issues, especially with recursing backends.

The big news is that wildcard CNAMEs are now supported, an oft requested feature and nearly the only part in which PDNS differed from BIND in authoritative capabilities.

If you were seeing signal 6 errors in PDNS causing reloads and intermittent service disruptions, please upgrade to this version.

For operators of PowerDNS Express trying to host .DE domains, the very special **soa-serial-offset** feature has been added to placate the new DENIC requirement that the SOA serial be at least six digits. PowerDNS Express uses the SOA serial as an actual serial and not to insert dates and hence often has single digit soa serial numbers, causing big problems with .DE redelegations.

Bugs fixed:

- Malformed or shortened TCP recursion queries would cause a signal 6 and a reload. Same for EOF from the TCP recursing backend. Thanks to Simon Kirby and Mike Benoit of NetNation for helping debug this.
- Timeouts on the TCP recursing backend were far too long, leading to possible exhaustion of TCP resolving threads.
- **pdns_control purge domain** accidentally cleaned all packets with that name as a prefix. Thanks to Simon Kirby for spotting this.
- Improved exception error logging - in some circumstances PDNS would not properly log the cause of an exception, which hampered problem resolution.

New features:

- Wildcard CNAMEs now work as expected!

- **pdns_control purge** can now also purge based on suffix, allowing operators to purge an entire domain from the packet cache instead of only specific records. See also Section B.1.1 Thanks to Mike Benoit for this suggestion.
- **soa-serial-offset** for installations with small SOA serial numbers wishing to register .DE domains with DENIC which demands six-figure SOA serial numbers. See also Chapter 15.

1.3.20. Version 2.1

This is a somewhat bigger release due to pressing demands from customers. An upgrade is advised for installations using Recursion. If you are using recursion, it is vital that you are aware of changes in semantics. Basically, local data will now override data in your recursing backend under most circumstances. Old behaviour can be restored by turning **lazy-recursion** off.

Developers: this version has a new pdns-2.1 development kit, available on http://downloads.powerdns.com/releases/dev (http://downloads.powerdns.com/releases/dev). See also Appendix C.

Warning

Most users will run a static version of PDNS which has no dependencies on external libraries. However, some may need to run the dynamic version. This warning applies to these users.

To run the dynamic version of PDNS, which is needed for backend drivers which are only available in source form, gcc 3.0 is required. RedHat 7.2 comes with gcc 3.0 as an optional component, RedHat 7.3 does not. However, the RedHat 7.2 Update gcc rpms install just fine on RedHat 7.3. For Debian, we suggest running 'woody' and installing the g++-3.0 package. We expect to release a FreeBSD dynamic version shortly.

Bugs fixed:

- RPM releases sometimes overwrote previous configuration files. Thanks to Jorn Ekkelenkamp of Hubris/ISP Services for reporting this.
- TCP recursion sent out overly large responses due to a byteorder mistake, confusing some clients. Thanks to the capable engineers of NetNation for bringing this to our attention.
- TCP recursion in combination with a recursing backend on a non-standard port did not work, leading to a non-functioning TCP listener. Thanks to the capable engineers of NetNation for bringing this to our attention.

Unexpected behaviour:

- Wildcard URL records where not implemented because they are a performance penalty. To turn these on, enable **wildcard-url** in the configuration.
- Unlike other nameservers, local data did not override the internet for recursing queries. This has mostly been brought into conformance with user expectations. If a recursive question can be answered entirely from local data, it is. To restore old behaviour, disable **lazy-recursion**. Also see Chapter 11.

Features:

- Oracle support has been tuned, leading to the first public release of the Oracle backend. Zone2sql now outputs better SQL and the backend is now fully documented. Furthermore, the queries are compatible with the PowerDNS XML-RPC product, allowing PowerDNS express to run off Oracle. See Section A.6.
- Zone2sql now accepts --transactions to wrap zones in a transaction for PostgreSQL and Oracle output. This is a major speedup and also makes for better isolation of inserts. See Section 10.1.
- **pdns_control** now has the ability to purge the PowerDNS cache or parts of it. This enables operators to raise the TTL of the Packet Cache to huge values and only to invalidate the cache when changes are made. See also Chapter 9 and Section B.1.1.

1.3.21. Version 2.0.1

Maintenance release, fixing three small issues.

Developers: this version is compatible with 1.99.11 backends.

- PowerDNS ignored the **logging-facility** setting unless it was specified on the commandline. Thanks to Karl Obermayer from WebMachine Technologies for noticing this.
- Zone2sql neglected to preserve 'slaveness' of domains when converting to the slave capable PostgreSQL backend. Thanks to Mike Benoit of NetNation for reporting this. Zone2sql now has a **--slave** option.
- SOA Hostmaster addresses with dots in them before the @-sign were mis-encoded on the wire.

1.3.22. Version 2.0

Two bugfixes, one stability/security related. No new features.

Developers: this version is compatible with 1.99.11 backends.

Bugfixes:

- zone2sql refused to work under some circumstances, taking 100% cpu and not functioning. Thanks to Andrew Clark and Mike Benoit for reporting this.
- Fixed a stability issue where malformed packets could force PDNS to reload. Present in all earlier 2.0 versions.

1.3.23. Version 2.0 Release Candidate 2

Mostly bugfixes, no really new features.

Developers: this version is compatible with 1.99.11 backends.

Bugs fixed:

- chroot() works again - 2.0rc1 silently refused to chroot. Thanks to Hub Dohmen for noticing this.
- setuid() and setgid() security features were silently not being performed in 2.0rc1. Thanks to Hub Dohmen for noticing this.
- MX preferences over 255 now work as intended. Thanks to Jeff Crowe for noticing this.
- IPv6 clients can now also benefit from the recursing backend feature. Thanks to Andy Furnell for proving beyond any doubt that this did not work.
- Extremely bogus code removed from DNS notification reception code - please test! Thanks to Jakub Jermar for working with us in figuring out just how broken this was.
- AXFR code improved to handle more of the myriad different zonetransfer dialects available. Specifically, interoperability with Bind 4 was improved, as well as Bind 8 in 'strict rfc conformance' mode. Thanks again for Jakub Jermar for running many tests for us. If your transfers failed with 'Unknown type 14!!' or words to that effect, this was it.

Features:

- Win32 version now has a zone2sql tool.
- Win32 version now has support for specifying how urgent messages should be before they go to the NT event log.

Remaining issues:

- One persistent report of the default 'chroot=./' configuration not working.
- One report of disable-axfr and allow-axfr-ips not working as intended.
- Support for relative paths in zones and in Bind configuration is not bug-for-bug compatible with bind yet.

1.3.24. Version 2.0 Release Candidate 1

The MacOS X release! A very experimental OS X 10.2 build has been added. Furthermore, the Windows version is now in line with Unix with respect to capabilities. The ODBC backend now has the code to function as both a master and a slave.

Developers: this version is compatible with 1.99.11 backends.

- Implemented native packet response parsing code, allowing Windows to perform AXFR and NS and SOA queries.
- This is the first version for which we have added support for Darwin 6.0, which is part of the forthcoming Mac OS X 10.2. Please note that although this version is marked RC1, that we have not done extensive testing yet. Consider this a technology preview.

  - The Darwin version has been developed on Mac OS X 10.2 (6C35). Other versions may or may not work.
  - Currently only the random, bind, mysql and pdns backends are included.
  - The menu based installer script does not work, you will have to edit pathconfig by hand as outlined in chapter 2.
  - On Mac OS X Client, PDNS will fail to start because a system service is already bound to port 53.

  This version is distributed as a compressed tar file. You should follow the generic UNIX installation instructions.

Bugs fixed:

- Zone2sql PostgreSQL mode neglected to lowercase $ORIGIN. Thanks to Maikel Verheijen of Ladot for spotting this.
- Zone2sql PostgreSQL mode neglected to remove a trailing dot from $ORIGIN if present. Thanks to Thanks to Maikel Verheijen of Ladot for spotting this.
- Zonefile parser was not compatible with bind when $INCLUDING non-absolute filenames. Thanks to Jeff Miller for working out how this should work.
- Bind configuration parser was not compatible with bind when including non-absolute filenames. Thanks to Jeff Miller for working out how this should work.
- Documentation incorrectly listed the Bind backend as 'slave capable'. This is not yet true, now labeled 'experimental'.

Windows changes. We are indebted to Dimitry Andric who educated us in the ways of distributing Windows software.

- `pdns.conf` is now read if available.
- Console version responds to ^c now.
- Default pdns.conf added to distribution
- Uninstaller missed several files, leaving remnants behind
- DLLs are now installed locally, with the pdns executable.
- pdns_control is now also available on Windows
- ODBC backend can now act as master and slave. Experimental.
- The example zone missed indexes and had other faults.
- A runtime DLL that is present on most windows systems (but not all!) was missing.

1.3.25. Version 1.99.12 Prerelease

The Windows release! See Chapter 3. Beware, windows support is still very fresh and untested. Feedback is very welcome.

Developers: this version is compatible with 1.99.11 backends.

- Windows 2000 codebase merge completed. This resulted in quite some changes on the Unix end of things, so this may impact reliability
- ODBC backend added for Windows. See Section A.9.
- IBM DB2 Universal Database backend available for Linux. See Section A.7.

- Zone2sql now understands $INCLUDE. Thanks to Amaze Internet for nagging about this
- The SOA Mininum TTL now has a configurable default (**soa-minimum-ttl**)value to placate the DENIC requirements.
- Added a limit on the simultaneous numbers of TCP connections to accept (**max-tcp-connections**). Defaults to 10.

Bugs fixed:

- When operating in virtual hosting mode (See Chapter 8), the additional init.d scripts would not function correctly and interface with other pdns instances.
- PDNS neglected to conserve case on answers. So a query for WwW.PoWeRdNs.CoM would get an answer listing the address of www.powerdns.com. While this did not confuse resolvers, it is better to conserve case. This has semantical concequences for all backends, which the documentation now spells out.
- PostgreSQL backend was case sensitive and returned only answers in case an exact match was found. The Generic PostgreSQL backend is now officially all lower case and zone2sql in PostgreSQL mode enforces this. Documentation has been been updated to reflect the case change. Thanks to Maikel Verheijen of Ladot for spotting this!
- Documentation bug - postgresql create/index statements created a duplicate index. If you've previously copy pasted the commands and not noticed the error, execute **CREATE INDEX rec_name_index ON records(name)** to remedy. Thanks to Jeff Miller for reporting this. This also lead to depressingly slow 'ANY' lookups for those of you doing benchmarks.

Features:

- pdns_control (see Section B.1.1) now opens the local end of its socket in `/tmp` instead of next to the remote socket (by default `/var/run`). This eases the way for allowing non-root access to pdns_control. When running chrooted (see Chapter 7), the local socket again moves back to `/var/run`.
- pdns_control now has a 'version' command. See Section B.1.1.

1.3.26. Version 1.99.11 Prerelease

This release is important because it is the first release which is accompanied by an Open Source Backend Development Kit, allowing external developers to write backends for PDNS. Furthermore, a few bugs have been fixed:

- Lines with only whitespace in zone files confused PDNS (thanks Henk Wevers)

- PDNS did not properly parse TTLs with symbolic sufixes in zone files, ie 2H instead of 7200 (thanks Henk Wevers)

1.3.27. Version 1.99.10 Prerelease

IMPORTANT: there has been a tiny license change involving free public webbased dns hosting, check out the changes before deploying!

PDNS is now feature complete, or very nearly so. Besides adding features, a lot of 'fleshing out' work is done now. There is an important performance bug fix which may have lead to disappointing benchmarks - so if you saw any of that, please try either this version or 1.99.8 which also does not have the bug.

This version has been very stable for us on multiple hosts, as was 1.99.9.

PostgreSQL users should be aware that while 1.99.10 works with the schema as presented in earlier versions, advanced features such as master or slave support will not work unless you create the new 'domains' table as well.

Bugs fixed:

- Wildcard AAAA queries sometimes received an NXDOMAIN error where they should have gotten an empty NO ERROR. Thanks to Jeroen Massar for spotting this on the .TK TLD!
- Do not disable the packetcache for 'recursion desired' packets unless a recursor was configured. Thanks to Greg Schueler for noticing this.
- A failing backend would not be reinstated. Thanks to 'Webspider' for discovering this problem with PostgreSQL connections that die after prolonged inactivity.
- Fixed loads of IPv6 transport problems. Thanks to Marco Davids and others for testing. Considered ready for production now.
- **Zone2sql** printed a debugging statement on range $GENERATE commands. Thanks to Rene van Valkenburg for spotting this.

Features:

- PDNS can now act as a master, sending out notifications in case of changes and allowing slaves to AXFR. Big rewording of replication support, domains are now either 'native', 'master' or 'slave'. See Chapter 13 for lots of details.
- **Zone2sql** in PostgreSQL mode now populates the 'domains' table for easy master, slave or native replication support.

- Ability to disable those annoying Windows DNS Dynamic Update messages from appearing in the log. See `log-failed-updates` in Chapter 15.
- Ability to run on IPv6 transport only
- Logging can now happen under a 'facility' so all PDNS messages appear in their own file. See Section 6.3.
- Different OS releases of PDNS now get different install path defaults. Thanks to Mark Lastdrager for nagging about this and to Nero Imhard and Frederique Rijsdijk for suggesting saner defaults.
- Infrastructure for 'also-notify' statements added.

1.3.28. Version 1.99.9 Early Access Prerelease

This is again a feature and an infrastructure release. We are nearly feature complete and will soon start work on the backends to make sure that they are all master, slave and 'superslave' capable.

Bugs fixed:

- PDNS sometimes sent out duplicate replies for packets passed to the recursing backend. Mostly a problem on SMP systems. Thanks to Mike Benoit for noticing this.
- Out-of-bailiwick CNAMES (ie, a CNAME to a domain not in PDNS) caused a 'ServFail' packet in 1.99.8, indicating failure, leading to hosts not resolving. Thanks to Martin Gillstrom for noticing this.
- Zone2sql balked at zones editted under operating sytems terminating files with ^Z (Windows). Thanks Brian Willcott for reporting this.
- PostgreSQL backend logged the password used to connect. Now only does so in case of failure to connect. Thanks to 'Webspider' for noticing this.
- Debian unstable distribution wrongly depended on home compiled PostgreSQL libraries. Thanks to Konrad Wojas for noticing this.

Features:

- When operating as a slave, AAAA records are now supported in the zone. They were already supported in master zones.
- IPv6 transport support - PDNS can now listen on an IPv6 socket using the **local-ipv6** setting.
- Very silly randombackend added which appears in the documentation as a sample backend. See Appendix C.
- When transferring a slave zone from a master, out of zone data is now rejected. Malicious operators might try to insert bad records otherwise.
- 'Supermaster' support for automatic provisioning from masters. See Section 13.2.1.
- Recursing backend can now live on a non-standard (!=53) port. See Chapter 11.

- Slave zone retrieval is now queued instead of immediate, which scales better and is more resilient to temporary failures.
- **max-queue-length** parameter. If this many packets are queued for database attention, consider the situation hopeless and respawn.

Internal:

- SOA records are now 'special' and each backend can optionally generate them in special ways. PostgreSQL backend does so when operating as a slave.
- Writing backends is now a lot easier. See Appendix C.
- Added Bindbackend to internal regression tests, confirming that it is compliant.

1.3.29. Version 1.99.8 Early Access Prerelease

A lot of infrastructure work gearing up to 2.0. Some stability bugs fixed and a lot of new features.

Bugs fixed:

- Bindbackend was overly complex and crashed on some systems on startup. Simplified launch code.
- SOA fields were not always properly filled in, causing default values to go out on the wire
- Obscure bug triggered by malicious packets (we know who you are) in SOA finding code fixed.
- Magic serial number calculation contained a double free leading to instability.
- Standards violation, questions for domains for which PDNS was unauthoritative now get a SERVFAIL answer. Thanks to the IETF Namedroppers list for helping out with this.
- Slowly launching backends were being relaunched at a great rate when queries were coming in while launching backends.
- MySQL-on-unix-domain-socket on SMP systems was overwhelmed by the quick connection rate on launch, inserted a small 50ms delay.
- Some SMP problems appear to be compiler related. Shifted to GCC 3.0.4 for Linux.
- Ran ispell on documentation.

Feature enhancements:

- Recursing backend. See Chapter 11. Allows recursive and authoritative DNS on the same IP address.
- NAPTR support, which is especially useful for the ENUM/E.164 community.
- Zone transfers can now be allowed per netmask instead of only per IP address.

- Preliminary support for slave operation included. Only for the adventurous right now! See Section 13.2
- All record types now documented, see Chapter 17.

1.3.29.1. Known bugs

Wildcard CNAMES do not work as they do with bind.

Recursion sometimes sends out duplicate packets (fixed in 1.99.9 snapshots)

Some stability issues which are caught by the guardian

1.3.29.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend

1.3.30. Version 1.99.7 Early Access Prerelease

Named.conf parsing got a lot of work and many more bind configurations can now be parsed. Furthermore, error reporting was improved. Stability is looking good.

Bugs fixed:

- Bind parser got confused by filenames with underscores and colons.
- Bind parser got confused by spaces in quoted names
- FreeBSD version now stops and starts when instructed to do so.
- Wildcards were off by default, which violates standards. Now on by default.
- --oracle was broken in zone2sql

Feature enhancements:

- Line number counting goes on as it should when including files in named.conf
- Added --no-config to enable users to start the pdns daemon without parsing the configuration file.

- zone2sql now has --bare for unformatted output which can be used to generate insert statements for different database layouts
- zone2sql now has --gpgsql, which is an alias for --mysql, to output in a format useful for the default Generic PgSQL backend
- zone2sql is now documented.

1.3.30.1. Known bugs

Wildcard CNAMES do not work as they do with bind.

1.3.30.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend

Some of these features will be present in newer releases.

1.3.31. Version 1.99.6 Early Access Prerelease

This version is now running on dns-eu1.powerdns.net and working very well for us. But please remain cautious before deploying!

Bugs fixed:

- Webserver neglected to show log messages
- TCP question/answer miscounted multiple questions over one socket. Fixed misnaming of counter
- Packetcache now detects clock skew and times out entries
- named.conf parser now reports errors with line number and offending token
- Filenames in named.conf can now contain :

Feature enhancements:

- The webserver now by default does not print out configuration statements, which might contain database backends. Use **webserver-print-arguments** to restore the old behaviour.
- Generic PostgreSQL backend is now included. Still rather beta.

1.3.31.1. Known bugs

FreeBSD version does not stop when requested to do so.

Wildcard CNAMES do not work as they do with bind.

1.3.31.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend

Some of these features will be present in newer releases.

1.3.32. Version 1.99.5 Early Access Prerelease

The main focus of this release is stability and TCP improvements. This is the first release PowerDNS-the-company actually considers for running on its production servers!

Major bugs fixed:

- Zone2sql received a floating point division by zero error on named.confs with less than 100 domains.
- Huffman encoder failed without specific error on illegal characters in a domain
- Fixed huge memory leaks in TCP code.
- Removed further file descriptor leaks in guardian respawning code
- Pipebackend was too chatty.
- pdns_server neglected to close fds 0, 1 & 2 when daemonizing

Feature enhancements:

- bindbackend can be instructed not to check the ctime of a zone by specifying **bind-check-interval=0**, which is also the new default.
- **pdns_server --list-modules** lists all available modules.

Performance enhancements:

- TCP code now only creates a new database connection for AXFR.
- TCP connections timeout rather quickly now, leading to less load on the server.

1.3.32.1. Known bugs

FreeBSD version does not stop when requested to do so.

Wildcard CNAMES do not work as they do with bind.

1.3.32.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend, gpgsqlbackend

Some of these features will be present in newer releases.

1.3.33. Version 1.99.4 Early Access Prerelease

A lot of new named.confs can now be parsed, zone2sql & bindbackend have gained features and stability.

Major bugs fixed:

- Label compression was not always enabled, leading to large reply packets sometimes.
- Database errors on TCP server lead to a nameserver reload by the guardian.
- MySQL backend neglected to close its connection properly.
- BindParser miss parsed some IP addresses and netmasks.
- Truncated answers were also truncated on the packetcache, leading to truncated TCP answers.

Feature enhancements:

- Zone2sql and the bindbackend now understand the Bind $GENERATE{} syntax.
- Zone2sql can optionally gloss over non-existing zones with **--on-error-resume-next**.
- Zone2sql and the bindbackend now properly expand @ also on the right hand side of records.
- Zone2sql now sets a default TTL.
- DNS UPDATEs and NOTIFYs are now logged properly and sent the right responses.

Performance enhancements:

- 'Fancy records' are no longer queried for on ANY queries - this is a big speedup.

1.3.33.1. Known bugs

FreeBSD version does not stop when requested to do so.

Zone2sql refuses named.confs with less than 100 domains.

Wildcard CNAMES do not work as they do with bind.

1.3.33.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend, gpgsqlbackend

Some of these features will be present in newer releases.

1.3.34. Version 1.99.3 Early Access Prerelease

The big news in this release is the BindBackend which is now capable of parsing many more named.conf Bind configurations. Furthermore, PDNS has successfully parsed very large named.confs with large numbers of small domains, as well as small numbers of large domains (TLD).

Zone transfers are now also much improved.

Major bugs fixed:

- zone2sql leaked file descriptors on each domain, used wrong Bison recursion leading to parser stack overflows. This limited the amount of domains that could be parsed to 1024.
- zone2sql can now read all known zonefiles, with the exception of those containing $GENERATE
- Guardian relaunching a child lost two file descriptors
- Don't die on a connection reset by peer during zone transfer.
- Webserver does not crash anymore on ringbuffer resize

Feature enhancements:

- AXFR can now be disabled, and re-enabled per IP address

- --help accepts a parameter, will then show only help items with that prefix.
- zone2sql now accepts a --zone-name parameter
- BindBackend maturing - 9500 zones parsed in 3.5 seconds. No longer case sensitive.

Performance enhancements:

- Implemented RFC-breaking AXFR format (which is the industry standard). Zone transfers now zoom along at wirespeed (many megabits/s).

1.3.34.1. Known bugs

FreeBSD version does not stop when requested to do so.

BindBackend cannot parse zones with $GENERATE statements.

1.3.34.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend, gpgsqlbackend

Some of these features will be present in newer releases.

1.3.35. Version 1.99.2 Early Access Prerelease

Major bugs fixed:

- Database backend reload does not hang the daemon anymore
- Buffer overrun in local socket address initialisation may have caused binding problems
- setuid changed the uid to the gid of the selected user
- zone2sql doesn't coredump on invocation anymore. Fixed lots of small issues.
- Don't parse configuration file when creating configuration file. This was a problem with reinstalling.

Performance improvements:

- removed a lot of unnecessary gettimeofday calls
- removed needless select(2) call in case of listening on only one address
- removed 3 useless syscalls in the fast path

Having said that, more work may need to be done. Testing on a 486 saw packet rates in a simple setup (question/wait/answer/question..) improve from 200 queries/second to over 400.

Usability improvements:

- Fixed error checking in init.d script (**show**, **mrtg**)
- Added 'uptime' to the mrtg output
- removed further GNUisms from installer and init.d scripts for use on FreeBSD
- Debian package and apt repository, thanks to Wichert Akkerman.
- FreeBSD /usr/ports, thanks to Peter van Dijk (in progress).

Stability may be an issue as well as performance. This version has a tendency to log a bit too much which slows the nameserver down a lot.

1.3.35.1. Known bugs

Decreasing a ringbuffer on the website is a sure way to crash the daemon. Zone2sql, while improved, still has problems with a zone in the following format:

```
name          IN            A        1.2.3.4
              IN            A        1.2.3.5
```

To fix, add 'name' to the second line.

Zone2sql does not close filedescriptors.

FreeBSD version does not stop when requested via the init.d script.

1.3.35.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend, gpgsqlbackend
- fully functioning bindbackend - will try to parse named.conf, but probably fail

Some of these features will be present in newer releases.

1.3.36. Version 1.99.1 Early Access Prerelease

This is the first public release of what is going to become PDNS 2.0. As such, it is not of production quality. Even PowerDNS-the-company does not run this yet.

Stability may be an issue as well as performance. This version has a tendency to log a bit too much which slows the nameserver down a lot.

1.3.36.1. Known bugs

Decreasing a ringbuffer on the website is a sure way to crash the daemon. Zone2sql is very buggy.

1.3.36.2. Missing features

Features present in this document, but disabled or withheld from the current release:

- gmysqlbackend, oraclebackend, gpgsqlbackend
- fully functioning bindbackend - will not parse configuration files

Some of these features will be present in newer releases.

1.4. Security

As of the 8th of January 2003, no actual security problems with PowerDNS 2.9.4 or later are known about. This page will be updated with all bugs which are deemed to be security problems, or could conceivably lead to those. Any such notifications will also be sent to all PowerDNS mailinglists and BUGTRAQ.

All versions of PowerDNS before 2.9 are known to suffer from remote denial of service problems which can disrupt operation. Please upgrade to 2.9.4 as this page will only contain detailed security information from 2.9.4 onwards.

If you have a security problem to report, please email us at both <powerdns@powerdns.com> and at <ahu@ds9a.nl>. We adhere to the Rain Forest Puppy Full Disclosure Policy (RFPolicy) v2.0 (http://www.wiretrip.net/rfp/policy.html) and we ask you to do the same.

We remind PowerDNS users that under the terms of the GNU General Public License, PowerDNS comes with ABSOLUTELY NO WARRANTY. This license is included in the distribution and in this documentation, see Appendix E.

1.5. Acknowledgements

PowerDNS is grateful for the help of the following people or institutions:

- Dave Aaldering
- Wichert Akkerman
- Antony Antony
- Mike Benoit (NetNation Communication Inc.)
- Peter van Dijk
- Koos van den Hout
- Andre Koopal
- Eric Veldhuyzen
- Paul Wouters
- Thomas Wouters
- IETF Namedroppers mailinglist

Thanks!

(these people don’t share the blame for any errors or mistakes in powerdns - those are all ours)

Chapter 2. Installing on Unix

You will typically install PDNS > 2.9 via source or via a package. Earlier versions used a clumsy binary installer.

2.1. Possible problems at this point

At this point some things may have gone wrong. Typical errors include:

error while loading shared libraries: libstdc++.so.x: cannot open shared object file: No such file or directory

Errors looking like this indicate a mismatch between your PDNS distribution and your Unix operating system. Download the static PDNS distribution for your operating system and try again. Please contact <pdns@powerdns.com> if this is impractical.

2.2. Testing your install

After installing, it is a good idea to test the basic functionality of the software before configuring database backends. For this purpose, PowerDNS contains the 'bindbackend' which has a domain built in example.com, which is officially reserved for testing. To test, edit `pdns.conf` and add the following if not already present:

```
launch=bind
bind-example-zones
```

This configures powerdns to 'launch' the bindbackend, and enable the example zones. To fire up PDNS in testing mode, execute: **/etc/init.d/pdns monitor**, where you may have to substitute the location of your SysV init.d location you specified earlier. In monitor mode, the pdns process runs in the foreground and is very verbose, which is perfect for testing your install. If everything went all right, you can query the example.com domain like this:

```
host www.example.com 127.0.0.1
```

www.example.com should now have IP address 1.2.3.4. The **host** command can usually be found in the dnsutils package of your operating system. Alternate command is: **dig www.example.com A @127.0.0.1** or even **nslookup www.example.com 127.0.0.1**, although nslookup is not advised for DNS diagnostics.

- example.com SOA record
- example.com NS record pointing to ns1.example.com
- example.com NS record pointing to ns2.example.com
- example.com MX record pointing to mail.example.com

- example.com MX record pointing to mail1.example.com
- mail.example.com A record pointing to 4.3.2.1
- mail1.example.com A record pointing to 5.4.3.2
- ns1.example.com A record pointing to 4.3.2.1
- ns2.example.com A record pointing to 5.4.3.2
- host-0 to host-9999.example.com A record pointing to 2.3.4.5

When satisfied that basic functionality is there, type **QUIT** to exit the monitor mode. The adventurous may also type **SHOW *** to see some internal statistics. In case of problems, you will want to read the following section.

2.2.1. Typical errors

At this point some things may have gone wrong. Typical errors include:

binding to UDP socket: Address already in use

This means that another nameserver is listening on port 53 already. You can resolve this problem by determining if it is safe to shutdown the nameserver already present, and doing so. If uncertain, it is also possible to run PDNS on another port. To do so, add **local-port=5300** to `pdns.conf`, and try again. This however implies that you can only test your nameserver as clients expect the nameserver to live on port 53.

binding to UDP socket: Permission denied

You must be superuser in order to be able to bind to port 53. If this is not a possibility, it is also possible to run PDNS on another port. To do so, add **local-port=5300** to `pdns.conf`, and try again. This however implies that you can only test your nameserver as clients expect the nameserver to live on port 53.

Unable to launch, no backends configured for querying

PDNS did not find the **launch=bind** instruction in pdns.conf.

Multiple IP addresses on your server, PDNS sending out answers on the wrong one
Massive amounts of 'recvfrom gave error, ignoring: Connection refused'

If you have multiple IP addresses on the internet on one machine, UNIX often sends out answers over another interface than which the packet came in on. In such cases, use **local-address** to bind to specific IP addresses, which can be comma separated. The second error comes from remotes disregarding answers to questions it didn't ask to that IP address and sending back ICMP errors.

2.3. Running PDNS on unix

PDNS is normally controlled via a SysV-style init.d script, often located in `/etc/init.d` or `/etc/rc.d/init.d`. This script accepts the following commands:

monitor

Monitor is a special way to view the daemon. It executes PDNS in the foreground with a lot of logging turned on, which helps in determining startup problems. Besides running in the foreground, the raw PDNS control socket is made available. All external communication with the daemon is normally sent over this socket. While useful, the control console is not an officially supported feature. Commands which work are: **QUIT**, **SHOW ***, **SHOW varname**, **RPING**.

start

Start PDNS in the background. Launches the daemon but makes no special effort to determine success, as making database connections may take a while. Use **status** to query success. You can safely run **start** many times, it will not start additional PDNS instances.

restart

Restarts PDNS if it was running, starts it otherwise.

status

Query PDNS for status. This can be used to figure out if a launch was successful. The status found is prefixed by the PID of the main PDNS process.

stop

Requests that PDNS stop. Again, does not confirm success. Success can be ascertained with the **status** command.

dump

Dumps a lot of statistics of a running PDNS daemon. It is also possible to single out specific variable by using the **show** command.

show variable

Show a single statistic, as present in the output of the **dump**.

mrtg

See the performance monitoring Chapter 6.

Chapter 3. Installing on Microsoft Windows

> **Note:** PowerDNS support for Windows is, as of 1.99.12, very recent and therefore quite 'beta'. For reliability, we currently advise the use of the Unix versions. Furthermore there is no support for master or slave operation in the ODBC backend, which is the only one provided currently. This will be fixed soon.

As of 1.99.12, PowerDNS supports Windows natively. PDNS can act as an NT service and works with any ODBC drivers you may have.

To install PowerDNS for Windows you should check if your PC meets the following requirements:

- A PC running Microsoft NT (with a recent servicepack and at least mdac 2.5), 2000 or XP.
- An ODBC source containing valid zone information (an example MS Access database is supplied in the form of `powerdns.mdb`).

If your system meets these requirements, download the installer from http://www.powerdns.com/pdns/. After downloading the file begin the installation procedure by starting `powerdns-VERSION.exe`.

After installing the software you should create a valid ODBC source. To do this you have open the ODBC sources dialog: `Start->Settings->Control Panel->Administrative Tools->Data Sources (ODBC)`.

We'll use the example zone database that is included in the installation to explain how to create a source.

When you are in the ODBC sources dialog you activate the `System DSN` tab.

> **Note:** It is important to create a System DSN instead of an User DNS, otherwise the ODBC backend cannot function.

Press `Add...`, then you have to select a driver.

Select `Microsoft Access Driver (*.mdb)`.

Use `PowerDNS` as the DSN name, you can leave the description empty.

Then press `Select...` to select the database (ie. `C:\Program Files\PowerDNS\powerdns.mdb`).

Press `Ok` and you should be done.

For more information, see Section A.9.

3.1. Configuring PDNS on Microsoft Windows

You can specify program parameters in the `pdns.conf` file which should be located in pdns directory (ie. `C:\Program Files\PowerDNS\`).

To see a list of available parameters you can run `pdns.exe --help`.

Note: A default configuration file has been supplied with the installation.

3.2. Running PDNS on Microsoft Windows

If you installed pdns on Windows NT, 2000 or XP you can run pdns as a service.

This is how to do it: Go to services (`Start->Settings->Control Panel->Administrative Tools->Services`) and locate `PDNS` (you should have registered the program as a NT service during the installation).

Double-click on `PDNS` and push the start button. You should now see a progress bar that gets to the end and see the status change to 'Started'.

This is the same as starting pdns like this: `pdns.exe --ntservice`

If you haven't registered pdns as a service during the installation you can do so from the commandline by starting pdns like this: `pdns.exe --register-service`

You can run pdns as a standard console program by using a command prompt or `Start->Run...` This way you can specify command-line parameters (see the documentation for commandline options).

If you chose to add a PowerDNS menu to the start menu during the installation you can start pdns using the pdns shortcut in that menu.

Chapter 4. Configure database connectivity

This chapter shows you how to configure the Generic MySQL backend, which we like a lot. But feel free to use any of the myriad other backends. This backend is called 'gmysql', and needs to be configured in `pdns.conf`. Add the following lines, adjusted for your local setup:

```
launch=gmysql
gmysql-host=127.0.0.1
gmysql-user=root
gmysql-dbname=pdnstest
```

Remove any earlier **launch** statements. Also remove the **bind-example-zones** statement as the **bind** module is no longer launched.

> **Warning**
>
> Make sure that you can actually resolve the hostname of your database without accessing the database! It is advised to supply an IP address here to prevent chicken/egg problems!

> **Warning**
>
> Be very very sure that you configure the *g*mysql backend and not the mysql backend. See Section A.5. If you use the 'mysql' backend things will only appear to work.

Now start PDNS using the monitor command:

```
# /etc/init.d/pdns monitor
(...)
15:31:30 PowerDNS 1.99.0 (Mar 12 2002, 15:00:28) starting up
15:31:30 About to create 3 backend threads
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdnstest'
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdnstest'
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdnstest'
```

This is as to be expected - we did not yet add anything to MySQL for PDNS to read from. At this point you may also see other errors which indicate that PDNS either could not find your MySQL server or was unable to connect to it. Fix these before proceeding.

General MySQL knowledge is assumed in this chapter, please do not interpret these commands as DBA advice!

4.1. Configuring MySQL

Connect to MySQL as a user with sufficient privileges and issue the following commands:

```
create table domains (
 id  INT auto_increment,
 name  VARCHAR(255) NOT NULL,
 master  VARCHAR(20) DEFAULT NULL,
 last_check  INT DEFAULT NULL,
 type  VARCHAR(6) NOT NULL,
 notified_serial INT DEFAULT NULL,
 account         VARCHAR(40) DEFAULT NULL,
 primary key (id)
)type=InnoDB;

CREATE UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
  id              INT auto_increment,
  domain_id       INT DEFAULT NULL,
  name            VARCHAR(255) DEFAULT NULL,
  type            VARCHAR(6) DEFAULT NULL,
  content         VARCHAR(255) DEFAULT NULL,
  ttl             INT DEFAULT NULL,
  prio            INT DEFAULT NULL,
  change_date     INT DEFAULT NULL,
  primary key(id)
)type=InnoDB;

CREATE INDEX rec_name_index ON records(name);
CREATE INDEX nametype_index ON records(name,type);
CREATE INDEX domain_id ON records(domain_id);

create table supermasters (
  ip VARCHAR(25) NOT NULL,
  nameserver VARCHAR(255) NOT NULL,
  account VARCHAR(40) DEFAULT NULL
);

GRANT SELECT ON supermasters TO pdns;
GRANT ALL ON domains TO pdns;
GRANT ALL ON records TO pdns;
```

Now we have a database and an empty table. PDNS should now be able to launch in monitor mode and display no errors:

```
  # /etc/init.d/pdns monitor
  (...)
```

```
15:31:30 PowerDNS 1.99.0 (Mar 12 2002, 15:00:28) starting up
15:31:30 About to create 3 backend threads
15:39:55 [gMySQLbackend] MySQL connection succeeded
15:39:55 [gMySQLbackend] MySQL connection succeeded
15:39:55 [gMySQLbackend] MySQL connection succeeded
```

A sample query sent to the database should now return quickly without data:

```
$ host www.test.com 127.0.0.1
www.test.com A record currently not present at localhost
```

And indeed, the control console now shows:

```
Mar 12 15:41:12 We're not authoritative for 'www.test.com', sending unauth normal respons
```

Now we need to add some records to our database:

```
# mysql pdnstest
mysql> INSERT INTO domains (name, type) values ('test.com', 'NATIVE');
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','localhost ahu@ds9a.nl 1','SOA',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','dns-us1.powerdns.net','NS',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','dns-eu1.powerdns.net','NS',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'www.test.com','199.198.197.196','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'mail.test.com','195.194.193.192','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'localhost.test.com','127.0.0.1','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','mail.test.com','MX',120,25);
```

If we now requery our database, **www.test.com** should be present:

```
$ host www.test.com 127.0.0.1
www.test.com         A 199.198.197.196

$ host -v -t mx test.com 127.0.0.1
Address: 127.0.0.1
Aliases: localhost

Query about test.com for record types MX
Trying test.com ...
Query done, 1 answer, authoritative status: no error
test.com             120 IN MX 25 mail.test.com
Additional information:
mail.test.com        120 IN A 195.194.193.192
```

To confirm what happened, issue the command **SHOW *** to the control console:

```
% show *
corrupt-packets=0,latency=0,packetcache-hit=2,packetcache-miss=5,packetcache-size=0,
```

```
qsize-a=0,qsize-q=0,servfail-packets=0,tcp-answers=0,tcp-queries=0,
timedout-packets=0,udp-answers=7,udp-queries=7,
%
```

The actual numbers will vary somewhat. Now enter **QUIT** and start PDNS as a regular daemon, and check launch status:

```
# /etc/init.d/pdns start
pdns: started
# /etc/init.d/pdns status
pdns: 8239: Child running
# /etc/init.d/pdns dump
pdns: corrupt-packets=0,latency=0,packetcache-hit=0,packetcache-miss=0,
packetcache-size=0,qsize-a=0,qsize-q=0,servfail-packets=0,tcp-answers=0,
tcp-queries=0,timedout-packets=0,udp-answers=0,udp-queries=0,
```

You now have a working database driven nameserver! To convert other zones already present, use the **zone2sql** described in Appendix A.

4.1.1. Common problems

Most problems involve PDNS not being able to connect to the database.

Can’t connect to local MySQL server through socket ’/tmp/mysql.sock’ (2)

Your MySQL installation is probably defaulting to another location for its socket. Can be resolved by figuring out this location (often `/var/run/mysqld.sock`), and specifying it in the configuration file with the **gmysql-socket** parameter.

Another solution is to not connect to the socket, but to 127.0.0.1, which can be achieved by specifying **gmysql-host=127.0.0.1**.

Host ’x.y.z.w’ is not allowed to connect to this MySQL server

These errors are generic MySQL errors. Solve them by trying to connect to your MySQL database with the MySQL console utility **mysql** with the parameters specified to PDNS. Consult the MySQL documentation.

Chapter 5. Dynamic resolution using the PipeBackend

Also included in the PDNS distribution is the PipeBackend. The PipeBackend is primarily meant for allowing rapid development of new backends without tight integration with PowerDNS. It allows end-users to write PDNS backends in any language. A perl sample is provided. The PipeBackend is also very well suited for dynamic resolution of queries. Example applications include DNS based loadbalancing, geo-direction, DNS based failover with low TTLs.

The Pipe Backend also has a separate chapter in the backends appendix, see Section A.1.

Note: The Pipe Backend currently does not function under FreeBSD 4.x and 5.x, probably due to unfavorable interactions between its threading implementation and the fork system call.

Interestingly, the Linux PowerDNS binary running under the Linuxulator on FreeBSD does work.

5.1. Deploying the PipeBackend with the BindBackend

Included with the PDNS distribution is the example.pl backend which has knowledge of the example.com zone, just like the BindBackend. To install both, add the following to your `pdns.conf`:

```
launch=pipe,bind
bind-example-zones
pipe-command=location/of/backend.pl
```

Please adjust the **pipe-command** statement to the location of the unpacked PDNS distribution. If your backend is slow, raise **pipe-timeout** from its default of 2000ms. Now launch PDNS in monitor mode, and perform some queries. Note the difference with the earlier experiment where only the BindBackend was loaded. The PipeBackend is launched first and thus gets queried first. The sample backend.pl script knows about:

- webserver.example.com A records pointing to 1.2.3.4, 1.2.3.5, 1.2.3.6
- www.example.com CNAME pointing to webserver.example.com
- MBOXFW (mailbox forward) records pointing to powerdns@example.com. See the smtpredir documentation for information about MBOXFW.

For more information about how to write exciting backends with the PipeBackend, see Appendix A.

Chapter 6. Logging & Monitoring PDNS performance

In a production environment, you will want to be able to monitor PDNS performance. For this purpose, currently two methods are available, the webserver and the init.d **dump**, **show** and **mrtg**, commands. Furthermore, PDNS can perform a configurable amount of operational logging. This chapter also explains how to configure syslog for best results.

6.1. Webserver

To launch the internal webserver, add a **webserver** statement to the pdns.conf. This will instruct the PDNS daemon to start a webserver on localhost at port 8081, without password protection. Only local users (on the same host) will be able to access the webserver by default. The webserver lists a lot of information about the PDNS process, including frequent queries, frequently failing queries, lists of remote hosts sending queries, hosts sending corrupt queries etc. The webserver does not allow remote management of the daemon. The following nameserver related configuration items are available:

webserver

If set to anything but 'no', a webserver is launched.

webserver-address

Address to bind the webserver to. Defaults to 127.0.0.1, which implies that only the local computer is able to connect to the nameserver! To allow remote hosts to connect, change to 0.0.0.0 or the physical IP address of your nameserver.

webserver-password

If set, viewers will have to enter this plaintext password in order to gain access to the statistics.

webserver-port

Port to bind the webserver to. Defaults to 8081.

6.2. Via init.d commands

As mentioned before, the init.d commands **dump**, **show** and **mrtg** fetch data from a running PDNS process. Especially **mrtg** is powerful - it outputs data in a format that is ready for processing by the MRTG graphing tool.

MRTG can make insightful graphics on the performance of your nameserver, enabling the operator to easily spot trends. MRTG can be found on http://people.ee.ethz.ch/~oetiker/webtools/mrtg/mrtg.html (http://people.ee.ethz.ch/~oetiker/webtools/mrtg/mrtg.html)

A sample mrtg.conf:

```
Interval: 5
WorkDir: /var/www/mrtg
WriteExpires: yes
Options[_]: growright,nopercent
XSize[_]: 600

#--------------------------------------------------------------

Target[udp-queries]: '/etc/init.d/pdns mrtg udp-queries udp-answers'
Options[udp-queries]: growright,nopercent,perminute
MaxBytes[udp-queries]: 600000
AbsMax[udp-queries]: 600000
Title[udp-queries]: Queries per minute
PageTop[udp-queries]: <H2>Queries per minute</H2>
WithPeak[udp-queries]: ymwd
YLegend[udp-queries]: queries/minute
ShortLegend[udp-queries]: q/m
LegendI[udp-queries]: udp-questions
LegendO[udp-queries]: udp-answers


Target[perc-failed]: '/etc/init.d/pdns mrtg udp-queries udp-answers'
Options[perc-failed]: growright,dorelpercent,perminute
MaxBytes[perc-failed]: 600000
AbsMax[perc-failed]: 600000
Title[perc-failed]: Queries per minute, with percentage success
PageTop[perc-failed]: <H2>Queries per minute, with percentage success</H2>
WithPeak[perc-failed]: ymwd
YLegend[perc-failed]: queries/minute
ShortLegend[perc-failed]: q/m
LegendI[perc-failed]: udp-questions
LegendO[perc-failed]: udp-answers


Target[packetcache-rate]: '/etc/init.d/pdns mrtg packetcache-hit udp-queries'
Options[packetcache-rate]: growright,dorelpercent,perminute
Title[packetcache-rate]: packetcache hitrate
MaxBytes[packetcache-rate]: 600000
AbsMax[packetcache-rate]: 600000
PageTop[packetcache-rate]: <H2>packetcache hitrate</H2>
WithPeak[packetcache-rate]: ymwd
YLegend[packetcache-rate]: queries/minute
ShortLegend[packetcache-rate]: q/m
LegendO[packetcache-rate]: total
LegendI[packetcache-rate]: hit
```

```
Target[packetcache-missrate]: ‘/etc/init.d/pdns mrtg packetcache-miss udp-queries‘
Options[packetcache-missrate]: growright,dorelpercent,perminute
Title[packetcache-missrate]: packetcache MISSrate
MaxBytes[packetcache-missrate]: 600000
AbsMax[packetcache-missrate]: 600000
PageTop[packetcache-missrate]: <H2>packetcache MISSrate</H2>
WithPeak[packetcache-missrate]: ymwd
YLegend[packetcache-missrate]: queries/minute
ShortLegend[packetcache-missrate]: q/m
LegendO[packetcache-missrate]: total
LegendI[packetcache-missrate]: MISS

Target[latency]: ‘/etc/init.d/pdns mrtg latency‘
Options[latency]: growright,nopercent,gauge
MaxBytes[latency]: 600000
AbsMax[latency]: 600000
Title[latency]: Query/answer latency
PageTop[latency]: <H2>Query/answer latency</H2>
WithPeak[latency]: ymwd
YLegend[latency]: usec
ShortLegend[latency]: usec
LegendO[latency]: latency
LegendI[latency]: latency

Target[recursing]: ‘/etc/init.d/pdns mrtg recursing-questions recursing-answers‘
Options[recursing]: growright,nopercent,gauge
MaxBytes[recursing]: 600000
AbsMax[recursing]: 600000
Title[recursing]: Recursive questions/answers
PageTop[recursing]: <H2>Recursing questions/answers</H2>
WithPeak[recursing]: ymwd
YLegend[recursing]: queries/minute
ShortLegend[recursing]: q/m
LegendO[recursing]: recursing-questions
LegendI[recursing]: recursing-answers
```

6.3. Operational logging using syslog

(**logging-facility** is available from 1.99.10 and onwards)

This chapter assumes familiarity with syslog, the unix logging device. PDNS logs messages with different levels. The more urgent the message, the lower the ’priority’. By default, PDNS will only log messages with an urgency of 3 or lower, but this can be changed using the **loglevel** setting in the configuration file. Setting it to 0 will eliminate all logging, 9 will log everything.

By default, logging is performed under the 'DAEMON' facility which is shared with lots of other programs. If you regard nameserving as important, you may want to have it under a dedicated facility so PDNS can log to its own files, and not clutter generic files.

For this purpose, syslog knows about 'local' facilities, numbered from LOCAL0 to LOCAL7. To move PDNS logging to LOCAL0, add **logging-facility=0** to your configuration.

Furthermore, you may want to have separate files for the differing prioties - preventing lower priority messages from obscuring important ones.

A sample syslog.conf might be:

```
local0.info                     -/var/log/pdns.info
local0.warn                     -/var/log/pdns.warn
local0.err                      /var/log/pdns.err
```

Where local0.err would store the really important messages. For performance and diskspace reasons, it is advised to audit your syslog.conf for statements also logging PDNS activities. Many syslog.confs have a '*.*' statement to /var/log/syslog, which you may want to remove.

For performance reasons, be especially certain that no large amounts of synchronous logging take place. Under Linux, this is indicated by filenames not starting with a '-' - indicating a synchronous log, which hurts performance.

Chapter 7. Security settings & considerations

7.1. Settings

PDNS has several options to easily allow it to run more securely. Most notable are the **chroot**, **setuid** and **setgid** options which can be specified.

7.1.1. Running as a less privileged identity

By specifying **setuid** and **setgid**, PDNS changes to this identity shortly after binding to the privileged DNS ports. These options are highly recommended. It is suggested that a separate identity is created for PDNS as the user 'nobody' is in fact quite powerful on most systems.

Both these parameters can be specified either numerically or as real names. You should set these parameters immediately if they are not set!

7.1.2. Jailing the process in a chroot

The **chroot** option secures PDNS to its own directory so that even if it should become compromised and under control of external influences, it will have a hard time affecting the rest of the system.

Even though this will hamper hackers a lot, chroot jails have been known to be broken.

When chrooting PDNS, take care that backends will be able to get to their files. Many databases need access to a UNIX domain socket which should live within the chroot. It is often possible to hardlink such a socket into the chroot dir.

The default PDNS configuration is best chrooted to `./`, which boils down to the configured location of the controlsocket.

This is achieved by adding the following to pdns.conf: **chroot=./**, and restarting PDNS.

7.2. Considerations

In general, make sure that the PDNS process is unable to execute commands on your backend database. Most database backends will only need SELECT privilege. Take care to not connect to your database as the 'root' or 'sa' user, and configure the chosen user to have very slight privileges.

Databases empathic-ally do not need to run on the same machine that runs PDNS! In fact, in benchmarks it has been discovered that having a separate database machine actually improves performance.

Separation will enhance your database security highly. Recommended.

Chapter 8. Virtual hosting

It may be advantageous to run multiple separate PDNS installations on a single host, for example to make sure that different customers cannot affect each others zones. PDNS fully supports running multiple instances on one host.

To generate additional PDNS instances, copy the init.d script `pdns` to `pdns-name`, where `name` is the name of your virtual configuration. Must not contain a - as this will confuse the script.

When you launch PDNS via this renamed script, it will seek configuration instructions not in `pdns.conf` but in `pdns-name.conf`, allowing for separate specification of parameters.

Be aware however that the init.d **force-stop** will kill all PDNS instances!

Chapter 9. Performance related settings

Different backends will have different characteristics - some will want to have more parallel instances than others. In general, if your backend is latency bound, like most relational databases are, it pays to open more backends.

This is done with the **distributor-threads** setting. Of special importance is the choice between 1 or more backends. In case of only 1 thread, PDNS reverts to unthreaded operation which may be a lot faster, depending on your operating system and architecture.

Another very important setting **cache-ttl**. PDNS caches entire packets it sends out so as to save the time to query backends to assemble all data. The default setting of 10 seconds may be low for high traffic sites, a value of 60 seconds rarely leads to problems.

Some PDNS operators set cache-ttl to many hours or even days, and use **pdns_control purge** to selectively or globally notify PDNS of changes made in the backend. Also look at the Query Cache described in this chapter. It may materially improve your performance.

To determine if PDNS is unable to keep up with packets, determine the value of the **qsize-q** variable. This represents the number of packets waiting for database attention. During normal operations the queue should be small.

If it is known that backends will not contain CNAME records, the **skip-cname** setting can be used to prevent the normally mandatory CNAME lookup that is needed at least once for each DNS query.

Much the same holds for the **wildcards** setting. On by default, each non-existent query will lead to a number of additional wildcard queries. If it is known that the backends do not contain wildcard records, performance can be improved by adding **wildcards=no** to `pdns.conf`.

Logging truly kills performance as answering a question from the cache is an order of magnitude less work than logging a line about it. Busy sites will prefer to turn **log-dns-details** and **log-failed-updates** off.

9.1. Packet Cache

PDNS by default uses the ’Packet Cache’ to recognise identical questions and supply them with identical answers, without any further processing. The default time to live is 10 seconds. It has been observed that the utility of the packet cache increases with the load on your nameserver.

Not all backends may benefit from the packetcache. If your backend is memory based and does not lead to context switches, the packetcache may actually hurt performance.

The size of the packetcache can be observed with **/etc/init.d/pdns show packetcache-size**

9.2. Query Cache

Besides entire packets, PDNS can also cache individual backend queries. Each DNS query leads to a number of backend queries, the most obvious additional backend query is the check for a possible CNAME. So, when a query comes in for the 'A' record for 'www.powerdns.com', PDNS must first check for a CNAME for 'www.powerdns.com'.

The Query Cache caches these backend queries, many of which are quite repetitive. PDNS only caches queries with no answer, or with exactly one. In the future this may be expanded but this lightweight solution is very simple and therefore fast.

Most gain is made from caching negative entries, ie, queries that have no answer. As these take little memory to store and are typically not a real problem in terms of speed-of-propagation, the default TTL for negative queries is a rather high 60 seconds.

This only is a problem when first doing a query for a record, adding it, and immediately doing a query for that record again. It may then take up to 60 seconds to appear. Changes to existing records however do not fall under the negative query ttl (**negquery-cache-ttl**), but under the generic **query-ttl** which defaults to 20 seconds.

The default values should work fine for many sites. When tuning, keep in mind that the Query Cache mostly saves database access but that the Packet Cache also saves a lot of CPU because 0 internal processing is done when answering a question from the Packet Cache.

Chapter 10. Migrating to PDNS

Before migrating to PDNS a few things should be considered.

PDNS is not a recursing nameserver on its own

If PDNS receives a question for which it is not authoritative, it can't go out on the net to figure out an answer. However, because many installations are expected to be both authoritative and recursing, PDNS can use a separate recursing backend to provide non-authoritative answers. See Chapter 11 for more details.

PDNS does not operate as a 'slave' or 'master' server with all backends

Only the Generic PostgreSQL, Generic MySQL and BIND backends have the ability to act as master or slave.

To migrate, the **zone2sql** tool is provided.

10.1. Zone2sql

Zone2sql parses Bind named.conf files and zonefiles and outputs SQL on standard out, which can then be fed to your database.

Zone2sql understands the Bind master file extension '$GENERATE' and will also honour '$ORIGIN' and '$TTL'.

For backends supporting slave operation (currently only the Generic PostgreSQL, Generic MySQL and BIND backend), there is also an option to keep slave zones as slaves, and not convert them to native operation.

By default, zone2sql outputs code suitable for the mysqlbackend, but it can also generate SQL for the Generic PostgreSQL, Generic MySQL and Oracle backends. The following commands are available:

--bare

Output in a bare format, suitable for further parsing. The output is formatted as follows:

```
domain_id<TAB>'qname'<TAB>'qtype'<TAB>'content'<TAB>prio<TAB>ttl
```

--gmysql

Output in format suitable for the default configuration of the Generic MySQL backend.

--gpgsql

Output in format suitable for the default configuration of the Generic PostgreSQL backend.

--help

List options.

--mysql

Output in format suitable for the default configuration of the MySQL backend. Default.

--named-conf=...

Parse this named.conf to find locations of zones.

--on-error-resume-next

Ignore missing files during parsing. Dangerous.

--oracle

Output in format suitable for the default configuration of the Generic Oracle backend.

--slave

Maintain slave status of zones listed in named.conf as being slaves. The default behaviour is to convert all zones to native operation.

--startid

Supply a value for the first domain_id generated. Defaults at 0.

--transactions

For Oracle and PostgreSQL output, wrap each domain in a transaction for higher speed and integrity.

--verbose

Be verbose during conversion.

--zone=...

Parse only this zone file. Conflicts with **--named-conf** parameter.

--zone-name=...

When parsing a single zone without $ORIGIN statement, set this as the zone name.

Chapter 11. Recursion

(only available from 1.99.8 and onwards, recursing component available since 2.9.5)

From 2.9.5 onwards, PowerDNS offers both authoritative nameserving capabilities and a recursive nameserver component. These two halves are normally separate but many users insist on combining both recursion and authoritative service on one IP address. This can be likened to running Apache and Squid both on port 80.

However, many sites want to do this anyhow and some with good reason. For example, a setup like this allows the creation of fake domains which only exist for local users. Such domains often don’t end on ".com" or ".org" but on ".intern" or ".name-of-isp".

PowerDNS can cooperate with either its own recursor or any other you have available to deliver recursive service on its port.

By specifying the **recursor** option in the configuration file, questions requiring recursive treatment will be handed over to the IP address specified. An example configuration might be **recursor=130.161.180.1**, which designates 130.161.180.1 as the nameserver to handle recursive queries.

As of 2.9.5, the recursing component of PowerDNS is a bit young and relatively untested but we hope people will want to use it anyhow. As an alternative, we highly advise the use of the DJBDNS dnscache (http://cr.yp.to/djbdns/dnscache.html).

Take care not to point **recursor** to PDNS, which leads to a very tight packet loop!

By specifying **allow-recursion**, recursion can be restricted to netmasks specified. The default is to allow recursion from everywhere. Example: **allow-recursion=192.168.0.0/24, 10.0.0.0/8, 1.2.3.4**.

11.1. Details

Questions carry a number of flags. One of these is called ’Recursion Desired’. If PDNS is configured to allow recursion, AND such a flag is seen, AND the IP address of the client is allowed to recurse via PDNS, then the packet may be handed to the recursing backend.

If a Recursion Desired packet arrives and PDNS is configured to allow recursion, but not to the IP address of the client, resolution will proceed as if the RD flag were unset and the answer will indicate that recursion was not available.

It is also possible to use a resolver living on a different port. To do so, specify a recursor like this: **recursor=130.161.180.1:5300**.

If the backend does not answer a question within a large amount of time, this is logged as 'Recursive query for remote 10.96.0.2 with internal id 0 was not answered by backend within timeout, reusing id'. This may happen when using 'BIND' as a recursor as it is prone to drop queries which it can't answer immediately.

To make sure that the local authoritative database overrides recursive information, PowerDNS first tries to answer a question from its own database. If that succeeds, the answer packet is sent back immediately without involving the recursor in any way.

Some packets, like those asking for MX records which are needed for SMTP transport of email, can be subject to 'additional processing'. This means that a recursing nameserver is obliged to try to add A records (IP addresses) for any of the mailservers mentioned in the packet, should it have these addresses available.

If PowerDNS encounters records needing such processing and finds that it does not have the data in its authoritative database, it will send an opportunistic quick query to the recursing component to see if it perhaps has such data. This question is worded such that the recursing nameserver should return immediately such as not to block the authoritative nameserver.

This marks a change from pre-2.9.5 behaviour where a packet was handed wholesale to the recursor in case it needed additional processing which could not proceed from the authoritative database.

Chapter 12. PowerDNS resolver/recursing nameserver

As of 2.9.4, a small recursor comes with PowerDNS. The algorithm is influenced by the works of Dan J. Bernstein although all mistakes are ours. Here are the current faults, so nobody can accuse us of false advertising:

- Only ignores stale cache entries, does not actually clean them up. May replace them with newer data, however.
- Only compiles on Linux and possibly Solaris. FreeBSD 4.x decided not to support the POSIX get/set/swapcontext functions. Bug your favorite FreeBSD kernel or libc maintainer for a fix, or ask him to port MTasker (see below) to your operating system.
- It does not do TCP yet, and may have big problems with truncated packets.

To compile, add **--enable-recursor** to configure and the file `pdns_recursor` will be compiled. To run on a different port, use **./syncres --local-port=53**. To bind to another address, use the **local-address** setting.

> **Note:** PowerDNS author bert hubert has the pdns recursor in production and browsing with it works for him. Furthermore, the LARTC mailinglist (2000 subscribers) is using the pdns recursing nameserver.

Good points:

- Uses MTasker (homepage (http://ds9a.nl/mtasker))
- Can handle thousands of concurrent questions
- Code is written linearly, sequentially, which means that there are no problems with 'query restart' or anything.
- Relies heavily on Standard C++ Library infrastructure, which makes for little code (406 core lines).
- Is very verbose in showing how recursion actually works.
- The algorithm is simple and quite nifty.

12.1. pdns_recursor settings

At startup, the recursing nameserver reads the file `recursor.conf` from the configuration directory, often `/etc/powerdns` or `/usr/local/etc`.

The following settings can be configured:

aaaa-additional-processing

If turned on, the recursor will attempt to add AAAA IPv6 records to questions for MX records and NS records. Can be quite slow as absence of these records in earlier answers does not guarantee their non-existance. Can double the amount of queries needed. Off by default.

config-dir

Directory where the configuration file can be found.

daemon

Operate in the background, which is the default.

local-address

Local IP address (singular) to bind to. Defaults to all addresses.

local-port

Local port (singular) to bind to. Defaults to 53.

quiet

Don’t log queries.

trace

If turned on, output impressive heaps of logging. May destroy performance under load.

12.2. Details

PowerDNS implements a very simple but effective nameserver. Care has been taken not to overload remote servers in case of overly active clients.

This is implemented using the ’throttle’. This accounts all recent traffic and prevents queries that have been sent out recently from going out again.

There are three levels of throttling.

- If a remote server indicates that it is lame for a zone, the exact question won't be repeated in the next 60 seconds.
- After 4 ServFail responses in 60 seconds, the query gets throttled too.
- 5 timeouts in 20 seconds also lead to query suppression.

12.3. Statistics

Every half our or so, the recursor outputs a line with statistics. More infrastructure is planned so as to allow for Cricket or MRTG graphs. To force the output of statistics, send the process a SIGUSR1. A line of statistics looks like this:

```
Feb 10 14:16:03 stats: 125784 questions, 13971 cache entries, 309 negative entries, 84% cac
```

This means that there are 13791 different names cached, which each may have multiple records attached to them. There are 309 items in the negative cache, items of which it is known that don't exist and won't do so for the near future. 84% of incoming questions could be answered without any additional queries going out to the net.

The outpacket/query ratio means that on average, 0.37 packets were needed to answer a question. Initially this ratio may be well over 100% as additional queries may be needed to actually recurse the DNS and figure out the addresses of nameservers.

Finally, 12% of queries were not performed because identical queries had gone out previously, saving load servers worldwide.

Chapter 13. Master/Slave operation & replication

PDNS offers full master and slave semantics for replicating domain information. Furthermore, PDNS can benefit from native database replication.

13.1. Native replication

Native replication is the default, unless other operation is specifically configured. Native replication basically means that PDNS will not send out DNS update notifications, nor will react to them. PDNS assumes that the backend is taking care of replication unaided.

MySQL replication has proven to be very robust and well suited, even over transatlantic connections between badly peering ISPs. Other PDNS users employ Oracle replication which also works very well.

To use native replication, configure your backend storage to do the replication and do not configure PDNS to do so.

13.2. Slave operation

On launch, PDNS requests from all backends a list of domains which have not been checked recently for changes. This should happen every '**refresh**' seconds, as specified in the SOA record. All domains that are unfresh are then checked for changes over at their master. If the SOA serial number there is higher, the domain is retrieved and inserted into the database. In any case, after the check the domain is declared 'fresh', and will only be checked again after '**refresh**' seconds have passed.

Warning

Slave support is OFF by default, turn it on by adding **slave** to the configuration. The same holds for master operation. Both can be on simultaneously.

PDNS also reacts to notifies by immediately checking if the zone has updated and if so, retransfering it.

All backends which implement this feature must make sure that they can handle transactions so as to not leave the zone in a half updated state. MySQL configured with either BerkeleyDB or InnoDB meets this

requirement, as do PostgreSQL and Oracle. The Bindbackend implements transaction semantics by renaming files if and only if they have been retrieved completely and parsed correctly.

Slave operation can also be programmed using several pdns_control commands, see Section B.1.1. The 'retrieve' command is especially useful as it triggers an immediate retrieval of the zone from the configured master.

13.2.1. Supermaster automatic provisioning of slaves

PDNS can recognize so called 'supermasters'. A supermaster is a host which is master for domains and for which we are to be a slave. When a master (re)loads a domain, it sends out a notification to its slaves. Normally, such a notification is only accepted if PDNS already knows that it is a slave for a domain.

However, a notification from a supermaster carries more persuasion. When PDNS determines that a notification comes from a supermaster and it is bonafide, PDNS can provision the domain automatically, and configure itself as a slave for that zone.

To enable this feature, a backend needs to know about the IP address of the supermaster, and how PDNS will be listed in the set of NS records remotely, and the 'account' name of your supermaster. There is no need to fill this out but it does help keep track of where a domain comes from.

13.3. Master operation

When operating as a master, PDNS sends out notifications of changes to slaves, which react to these notifications by querying PDNS to see if the zone changed, and transferring its contents if it has. Notifications are a way to promptly propagate zone changes to slaves, as described in RFC 1996.

Warning

Master support is OFF by default, turn it on by adding **master** to the configuration. The same holds for slave operation. Both can be on simultaneously.

Left open by RFC 1996 is who is to be notified - which is harder to figure out than it sounds. All slaves for this domain must receive a notification but the nameserver only knows the names of the slaves - not the IP addresses, which is where the problem lies. The nameserver itself might be authoritative for the name of its secondary, but not have the data available.

To resolve this issue, PDNS tries multiple tactics to figure out the IP addresses of the slaves, and notifies everybody. In contrived configurations this may lead to duplicate notifications being sent out, which shouldn't hurt.

Some backends may be able to detect zone changes, others may chose to let the operator indicate which zones have changed and which haven't. Consult the documentation for your backend to see how it processes changes in zones.

To help deal with slaves that may have missed notifications, or have failed to respond to them, several override commands are available via the pdns_control tool (Section B.1.1):

pdns_control notify **domain**

This instructs PDNS to notify all IP addresses it considers to be slaves of this domain.

pdns_control notify-host **domain ip-address**

This is truly an override and sends a notification to an arbitrary IP address. Can be used in 'also-notify' situations or when PDNS has trouble figuring out who to notify - which may happen in contrived configurations.

Chapter 14. Fancy records for seamless email and URL integration

PDNS also supports so called 'fancy' records. A Fancy Record is actually not a DNS record, but it is translated into one. Currently, two fancy records are implemented, but not very useful without additional unreleased software. For completeness, they are listed here. The software will become available later on and is part of the Express and PowerMail suite of programs.

These records imply extra database lookups which has a performance impact. Therefore fancy records are only queried for if they are enabled with the **fancy-records** command in `pdns.conf`.

MBOXFW

This record denotes an email forward. A typical entry looks like this:

```
support@yourdomain.com      MBOXFW        you@yourcompany.com
```

When PDNS encounters a request for an MX record for yourdomain.com it will, if fancy records are enabled, also check for the existence of an MBOXFW record ending on '@yourdomain.com', in which case it will hand out a record containing the configured **smtpredirector**. This server should then also be able to access the PDNS database to figure out where mail to support@yourdomain.com should go to.

URL

URL records work in much the same way, but for HTTP. A sample record:

```
yourdomain.com      URL        http://somewhere.else.com/yourdomain
```

A URL record is converted into an A record containing the IP address configured with the **urlredirector** setting. On that IP address a webserver should live that knows how to redirect yourdomain.com to http://somewhere.else.com/yourdomain.

Chapter 15. Index of all settings

All PDNS settings are listed here, excluding those that originate from backends, which are documented in the relevant chapters.

allow-axfr-ips=...

Behaviour pre 2.9.10: When not allowing AXFR (disable-axfr), DO allow from these IP addresses or netmasks.

Behaviour post 2.9.10: If set, only these IP addresses or netmasks will be able to perform AXFR.

allow-recursion=...

By specifying **allow-recursion**, recursion can be restricted to netmasks specified. The default is to allow recursion from everywhere. Example: **allow-recursion=192.168.0.0/24, 10.0.0.0/8, 1.2.3.4**.

cache-ttl=...

Seconds to store packets in the PacketCache. See Section 9.1.

chroot=...

If set, chroot to this directory for more security. See Chapter 7.

config-dir=...

Location of configuration directory (pdns.conf)

config-name=...

Name of this virtual configuration - will rename the binary image. See Chapter 8.

control-console=...

Debugging switch - don’t use.

daemon=...

Operate as a daemon

default-soa-name=...

name to insert in the SOA record if none set in the backend

disable-axfr=...

Do not allow zone transfers. Before 2.9.10, this could be overridden by allow-axfr-ips.

disable-tcp=...

Do not listen to TCP queries. Breaks RFC compliance.

distributor-threads=...

Default number of Distributor (backend) threads to start. See Chapter 9.

fancy-records=...

Process URL and MBOXFW records. See Chapter 14.

guardian | --guardian=yes | --guardian=no

Run within a guardian process. See Section B.2.

help

Provide a helpful message

launch=...

Which backends to launch and order to query them in. See Section B.3.

lazy-recursion=...

On by default as of 2.1. Checks local data first before recursing. See Chapter 11.

load-modules=...

Load this module - supply absolute or relative path. See Section B.3.

local-address=...

Local IP address to which we bind. You can specify multiple addresses separated by commas or whitespace. It is highly advised to bind to specific interfaces and not use the default 'bind to any'. This causes big problems if you have multiple IP addresses. Unix does not provide a way of figuring out what IP address a packet was sent to when binding to any.

local-port=...

The port on which we listen. Only one port possible.

log-failed-updates=...

If set to 'no', failed Windows Dynamic Updates will not be logged.

log-dns-details=...

If set to 'no', informative-only DNS details will not even be sent to syslog, improving performance. Available from 2.5 and onwards.

logging-facility=...

If set to a a digit, logging is performed under this LOCAL facility. See Section 6.3. Available from 1.99.9 and onwards.

loglevel=...

Amount of logging. Higher is more. Do not set below 3

max-queue-length=...

If this many packets are waiting for database attention, consider the situation hopeless and respawn.

module-dir=...

Default directory for modules. See Section B.3.

negquery-cache-ttl=...

Seconds to store queries with no answer in the Query Cache. See Section 9.2.

no-config

Do not attempt to read the configuration file.

out-of-zone-additional-processing | --out-of-zone-additional-processing=yes | --out-of-zone-additional-processing=no

Do out of zone additional processing

query-cache-ttl=...

Seconds to store queries with an answer in the Query Cache. See Section 9.2.

queue-limit=...

Maximum number of miliseconds to queue a query. See Chapter 9.

query-local-address=...

The IP address to use as a source address for sending queries. Useful if you have multiple IPs and pdns is not bound to the IP address your operating system uses by default for outgoing packets.

query-logging | query-logging=yes | query-logging=no

Hints to a backend that it should log a textual representation of queries it performs. Can be set at runtime.

recursive-cache-ttl=...

Seconds to store recursive packets in the PacketCache. See Section 9.1.

recursor=...

If set, recursive queries will be handed to the recursor specified here. See Chapter 11.

setgid=...

If set, change group id to this gid for more security. See Chapter 7.

setuid=...

If set, change user id to this uid for more security. See Chapter 7.

skip-cname | --skip-cname=yes | --skip-cname=no

Do not perform CNAME indirection for each query. Has performance implications. See Chapter 7.

slave-cycle-interval=60

Schedule slave up-to-date checks of domains whose status is unknown every .. seconds. See Chapter 14.

smtpredirector=...

Our smtpredir MX host. See Chapter 14.

soa-serial-offset=...

If your database contains single-digit SOA serials and you need to host .DE domains, this setting can help placate their 6-digit SOA serial requirements. Suggested value is to set this to 1000000 which adds 1000000 to all SOA Serials under that offset.

socket-dir=...

Where the controlsocket will live. See Section B.1.

strict-rfc-axfrs | --strict-rfc-axfrs=yes | --strict-rfc-axfrs=no

Perform strictly RFC conformant AXFRs, which are slow, but needed to placate some old client tools.

urlredirector=...

Where we send hosts to that need to be url redirected. See Chapter 14.

webserver | --webserver=yes | --webserver=no

Start a webserver for monitoring. See Chapter 6.

webserver-address=...

IP Address of webserver to listen on. See Chapter 6.

webserver-password=...

Password required for accessing the webserver. See Chapter 6.

webserver-port=...

Port of webserver to listen on. See Chapter 6.

wildcard-url=...

Check for wildcard URL records.

wildcards=...

Honor wildcards in the database. On by default. Turning this off has performance implications, see Chapter 9.

Chapter 16. Index of all internal metrics

16.1. Counters & variables

A number of counters and variables are set during PDNS operation. These can be queried with the init.d **dump**, **show** and **mrtg** commands, or viewed with the webserver.

corrupt-packets
: Number of corrupt packets received

latency
: Average number of microseconds a packet spends within PDNS

packetcache-hit
: Number of packets which were answered out of the cache

packetcache-miss
: Number of times a packet could not be answered out of the cache

packetcache-size
: Amount of packets in the packetcache

qsize-a
: Size of the queue before the transmitting socket.

qsize-q
: Number of packets waiting for database attention

servfail-packets
: Amount of packets that could not be answered due to database problems

tcp-answers
: Number of answers sent out over TCP

tcp-questions
: Number of questions received over TCP

timedout-questions
: Amount of packets that were dropped because they had to wait too long internally

udp-answers

Number of answers sent out over UDP

udp-questions

Number of questions received over UDP

16.1.1. Ring buffers

Besides counters, PDNS also maintains the ringbuffers. A ringbuffer records events, each new event gets a place in the buffer until it is full. When full, earlier entries get overwritten, hence the name 'ring'.

By counting the entries in the buffer, statistics can be generated. These statistics can currently only be viewed using the webserver and are in fact not even collected without the webserver running.

The following ringbuffers are available:

Log messages (logmessages)

All messages logged

Queries for existing records but for a type we don't have (noerror-queries)

Queries for, say, the AAAA record of a domain, when only an A is available. Queries are listed in the following format: name/type. So an AAA query for pdns.powerdns.com looks like pdns.powerdns.com/AAAA.

Queries for non-existing records within existing domains(nxdomain-queries)

If PDNS knows it is authoritative over a domain, and it sees a question for a record in that domain that does not exist, it is able to send out an authoritative 'no such domain' message. Indicates that hosts are trying to connect to services really not in your zone.

UDP queries received (udp-queries)

All UDP queries seen.

Remote server IP addresses (remotes)

Hosts querying PDNS. Be aware that UDP is anonymous - person A can send queries that appear to be coming from person B.

Remotes sending corrupt packets (remote-corrupts)

Hosts sending PDNS broken packets, possibly meant to disrupt service. Be aware that UDP is anonymous - person A can send queries that appear to be coming from person B.

Remotes querying domains for which we are not auth (remote-unauth)

> It may happen that there are misconfigured hosts on the internet which are configured to think that a PDNS installation is in fact a resolving nameserver. These hosts will not get useful answers from PDNS. This buffer lists hosts sending queries for domains which PDNS does not know about.

Queries that could not be answered due to backend errors (servfail-queries)

> For one reason or another, a backend may be unable to extract answers for a certain domain from its storage. This may be due to a corrupt database or to inconsistent data. When this happens, PDNS sends out a 'servfail' packet indicating that it was unable to answer the question. This buffer shows which queries have been causing servfails.

Queries for domains that we are not authoritative for (unauth-queries)

> If a domain is delegated to a PDNS instance, but the backend is not made aware of this fact, questions come in for which no answer is available, nor is the authority. Use this ringbuffer to spot such queries.

Chapter 17. Supported record types and their storage

This chapter lists all record types PDNS supports, and how they are stored in backends. The list is mostly alphabetical but some types are grouped.

A

The A record contains an IP address. It is stored as a decimal dotted quad string, for example: ’213.244.168.210’.

AAAA

The AAAA record contains an IPv6 address. An example: ’3ffe:8114:2000:bf0::1’.

CNAME

The CNAME record specifies the canonical name of a record. It is stored plainly. Like all other records, it is not terminated by a dot. A sample might be ’webserver-01.yourcompany.com’.

HINFO

Hardware Info record, used to specify CPU and operating system. Stored with a single space separating these two, example: ’i386 Linux’.

MX

The MX record specifies a mail exchanger host for a domain. Each mail exchanger also has a priority or preference. This should be specified in the separate field dedicated for that purpose, often called ’prio’.

NAPTR

Naming Authority Pointer, RFC 2915. Stored as follows:

```
’100  50  "s"  "z3950+I2L+I2C"     ""  _z3950._tcp.gatech.edu’.
```

The fields are: order, preference, flags, service, regex, replacement. Note that the replacement is not enclosed in quotes, and should not be. The replacement may be omitted, in which case it is empty. See also RFC 2916 for how to use NAPTR for ENUM (E.164) purposes.

NS

Nameserver record. Specifies nameservers for a domain. Stored plainly: ’ns1.powerdns.com’, as always without a terminating dot.

PTR

Reverse pointer, used to specify the host name belonging to an IP or IPv6 address. Name is stored plainly: ’www.powerdns.com’. As always, no terminating dot.

RP

Responsible Person record, as described in RFC 1183. Stored with a single space between the mailbox name and the more-information pointer. Example ’peter.powerdns.com peter.people.powerdns.com’, to indicate that peter@powerdns.com is responsible and that more information about peter is available by querying the TXT record of peter.people.powerdns.com.

SOA

The Start of Authority record is one of the most complex available. It specifies a lot about a domain: the name of the master nameserver (’the primary’), the hostmaster and a set of numbers indicating how the data in this domain expires and how often it needs to be checked. Further more, it contains a serial number which should rise on each change of the domain.

The stored format is:

```
primary hostmaster serial refresh retry expire default_ttl
```

Besides the primary and the hostmaster, all fields are numerical. PDNS has a set of default values:

Table 17-1. SOA fields

primary	**default-soa-name** configuration option
hostmaster	hostmaster@domain-name
serial	0
refresh	10800 (3 hours)
retry	3600 (1 hour)
expire	604800 (1 week)
default_ttl	3600 (1 hour)

The fields have complicated and sometimes controversial meanings. The ’serial’ field is special. If left at 0, the default, PDNS will perform an internal list of the domain to determine highest change_date field of all records within the zone, and use that as the zone serial number. This means that the serial number is always raised when changes are made to the zone, as long as the change_date field is being set.

SRV

SRV records can be used to encode the location and port of services on a domain name. When encoding, the priority field is used to encode the priority. For example, ’_ldap._tcp.dc._msdcs.conaxis.ch SRV 0 100 389 mars.conaxis.ch’ would be encoded with 0 in the priorit field and ’100 389 mars.conaxis.ch’ in the tontent field.

TXT

The TXT field can be used to attach textual data to a domain. Text is stored plainly.

Chapter 18. HOWTO & Frequently Asked Questions

This chapter contains a number of FAQs and HOWTOs.

18.1. Getting support, free and paid FAQ

PowerDNS is an open source program so you may get help from the PowerDNS users' community or from its authors. You may also help others (please do).

Some users may not have experience in interacting with developers or the open source community. This FAQ is to be considered MANDATORY READING before asking us for help.

Q: Help!

A: Please try harder. Specifically, before people will be able to help you, they need to know a lot about your system. Things you may find irrelevant. But, as you have a problem, you are not in a good position to know what is relevant and what not.

Q: I have a question, what details should I supply?

A: Start out with stating what you think should be happening. Quite often, wrong expectations are the actual proble. Furthermore, which database backend you use, your operating system, which version of PowerDNS you use and where you got it from (RPM, .DEB, tar.gz). If you compiled it yourself, what were the ./configure parameters.

In the Open Source community, not supplying vital details is interpreted as a lack of respect for those willing to take time to answer your questions!

If at *all* possible, supply the actual name of your domain and the IP address of your server(s).

Q: Where should I send my question?

A: To a mailinglist. Do not mail the authors directly unless you previously entered a support contract with them! For subscription details, see the mailinglists page (http://mailman.powerdns.com/mailman/admin/).

Questions about using PowerDNS should be sent to the pdns-users list, questions about compiler errors or feature requests to pdns-dev.

Before posting, read all FAQs and tell people you did.

Q: I'm special, I don't email to mailinglists!

We're special too, and we ask you to mail the mailinglists. If you need privacy, consider entering a support relationship with us, in which case you can email <support@powerdns.com>.

18.2. Using and Compiling PowerDNS FAQ

In the course of compiling and using PowerDNS, many questions may arise. Here are some we've heard earlier or questions we expect people may have. Please read this list before mailing us!

Q: Can I launch multiple backends simultaneously?

A: You can. This might for example be useful to keep an existing BIND configuration around but to store new zones in, say MySQL. The syntax to use is 'launch=bind,gmysql'.

Q: Which backend should I use? There are so many!

A: If you have no external constraints, the Generic MySQL (gmysql) and Generic PostgreSQL (gpgsql) ones are probably the most used and complete. By all means do not use the non-generic MySQL backend, which is deprecated and only available for older installations.

The Oracle backend also has happy users, we know of no deployments of the DB2 backend. The BIND backend is pretty capable too in fact, but many prefer a relational database.

Q: I try to launch the pgmysqlbackend and it can't find it!

A: You did not read the changelog, nor the README. The 'pgmysql' backend is no more and has been split into the gmysql and gpgsql backends, with the common code residing within PowerDNS itself.

Q: PowerDNS compiles under OpenBSD, but crashes immediately, now what?

A: Reasons behind this are somewhat unclear but we hear they go away if you use a more recent compiler. Let us know on <pdns-dev@mailman.powerdns.com>. See also here (http://www.codeninja.nl/openbsd/powerdns/).

Q: I'm trying to build from CVS but I get lots of weird errors!

A: Read the 'HACKING' file, it lists the build requirements (mostly autoconf, automake, libtool). In many cases, it may be easier to build from the source distribution though.

Q: I'm on Solaris 7 and AAAA records do not work

A: Indeed, and this is pretty sad. Either upgrade to Solaris 8 or convince people to write the replacement functions needed to encode AAAA if the host operating system does not offer them.

Q: When compiling I get errors about 'sstream' and 'ostringstream', or BITSPERCHAR

A: Your gcc is too old. Versions 2.95.2 and older are not supported. Many distributions have improved gcc 2.95.2 with an ostringstream implementation, in which case their 2.95.2 is also supported. We like gcc 3.2.1 best.

Q: Ok, I've installed gcc 3.2.1 but now the gpgsql backend won't link

A: Sadly, the gcc C++ on-disk object format has changed a few times since the 2.95 days. This means that gcc 3.2.1 cannot link against libpq++.so compiled with 2.95. The trick is to recompile PostgreSQL with 3.2.1 too and have it install in a separate location. Then reconfigure the pdns compile to look there, with **./configure --with-pgsql-lib=/opt/postgresql-with-3.2.1/lib**

Q: I've installed PostgreSQL 7.3 but it has no libpq++.so

A: As of 7.3, libpq++ has been split out of the main PostgreSQL distribution. See here (http://gborg.postgresql.org/). It would in fact be a great idea to move the gpgsql backend to the C interface instead of the C++ one. On Debian 'Sid', libpq++.so hides in the libpqpp-dev package.

Q: PowerDNS crashes when I install the pdns-static .deb on Debian SID

A: Indeed. Install the .debs that come with Debian or recompile PowerDNS yourself. If not using MySQL, the crashes will go away if you remove setuid and setgid statements from the configuration.

Q: Why don't my slaves act on notifications and transfer my updated zone?

A: Raise the serial number of your zone. In most backends, this is the first digit of the SOA contents field. If this number is lower to equal to that on a slave, it will not consider your zone updated.

Q: Master or Slave support is not working, PDNS is not picking up changes

A: The Master/Slave apparatus is off by default. Turn it on by adding a **slave** and/or **master** statement to the configuration file. Also, check that the configured backend is master or slave capable.

Q: My masters won't allow PowerDNS to access zones as it is using the wrong local IP address

A: Mark Bergsma contributed the query-local-address setting to tell PowerDNS which local IP address to use.

Q: I compiled PowerDNS myself and I see weird problems, especially on SMP

A: There are known issues between gcc <3.2 and PowerDNS on Linux SMP systems. The exact cause is not known but moving to our precompiled version always fixes the problems. If you compile yourself, use a recent gcc!

Q: PowerDNS does not answer queries on all my IP addresses and I've ignored the warning I got about that at startup

A: Please don't ignore what PowerDNS says to you. Furthermore, read Chapter 15 about the **local-address** setting, and use it to specify which IP addresses PowerDNS should listen on.

Q: Can I use a MySQL database with the Windows version of PowerDNS?

A: You can. MySQL support is supplied through the ODBC backend, which is compiled into the main binary. So if you want to use MySQL you can change the pdns.conf file, which is located in

the PowerDNS for Windows directory, to use the correct ODBC data sources. If you don't know how to use ODBC with MySQL:

- Download MyODBC from http://www.mysql.com/
- Install the MySQL ODBC driver.

Then you can follow the instructions located in Chapter 3. But instead of selecting the Microsoft Access Driver you select the MySQL ODBC Driver and configure it to use your MySQL database.

Note: For other databases for which an ODBC driver is available, the procedure is the same as this example.

18.3. Backend developer HOWTO

Writing backends without access to the full PDNS source means that you need to write code that can be loaded by PDNS at runtime. This in turn means that you need to use the same compiler that we do. For linux, this is currently GCC 3.0.4, although any 3.0.x compiler is probably fine. In tests, even 3.1 works.

For FreeBSD we use GCC 2.95.2.

Furthermore, your pdns_server executable must be dynamically linked. The default .rpm PDNS contains a static binary so you need to retrieve the dynamic rpm or the dynamic tar.gz or the Debian unstable ('Woody') deb. FreeBSD dynamic releases are forthcoming.

Q: Will PDNS drivers work with other PDNS versions than they were compiled for?

A: 'Probably'. We make no guarantees. Efforts have been made to keep the interface between the backend and PDNS as thin as possible. For example, a backend compiled with the 1.99.11 backend development kit works with 1.99.10. But don't count on it. We will notify when we think an incompatible API change has occured but you are best off recompiling your driver for each new PDNS release.

Q: What is in that DNSPacket * pointer passed to lookup!

A: For reasons outlined above, you should treat that pointer as opaque and only access it via the `getRemote()` functions made available and documented above. The DNSPacket class changes a lot and this level of indirection allows for greater changes to be made without changing the API to the backend coder.

Q: How is the PowerDNS Open Source Backend Development Kit licensed?

A: MIT X11, a very liberal license permitting basically everything.

Q: Can I release the backend I wrote?

A: Please do! If you tell us about it we will list you on our page.

Q: Can I sell backends I wrote?

A: You can. Again, if you tell us about them we will list your backend on the site. You can keep the source of your backend secret if you want, or you can share it with the world under any license of your chosing.

Q: Will PowerDNS use my code in the PDNS distribution?

A: If your license permits it and we like your backend, we sure will. If your license does not permit it but we like your backend anyway we may contact you.

Q: My backend compiles but when I try to load it, it says 'undefined symbol: _Z13BackendMakersv'

A: Your pdns_server binary is static and cannot load a backend driver at runtime. Get a dynamic version of pdns, or complain to pdns@powerdns.com if one isn't available. To check what kind of binary you have, execute 'file $(which pdns_server)'.

Q: My backend compiles but when I try to load it, it says 'undefined symbol: BackendMakers__Fv'

A: You compiled with the wrong GCC. Use GCC 3.x for Linux, 2.95.x for FreeBSD. You may want to change g++ to g++-3.0 in the Makefile, or change your path so that 3.x is used.

Q: I downloaded a dynamic copy of pdns_server but it doesn't run, even without my backend

A: Run 'ldd' on the pdns_server binary and figure out what libraries you are missing. Most likely you need to install gcc 3.0 libraries, RedHat 7.1 and 7.2 have packages available, Debian installs these by default if you use the 'unstable deb' of PDNS.

Q: What I want can't be done from a backend - I need the whole PDNS source

A: If you require the source, please contact us (pdns@powerdns.com). All commercial licensees receive the source, for others we may grant exceptions.

Q: What is this 'AhuException' I keep reading about?

A: This name has historical reasons and has no significance (http://ds9a.nl).

Q: I need a backend but I can't write it, can you help?

A: Yes, we also do custom development. Contact us at pdns@powerdns.com.

18.4. About PowerDNS.COM BV, 'the company'

As of 25 November 2002, the PowerDNS nameserver and its modules are open source. This has led to a lot of questions on the future of both PowerDNS, the company and the products. This FAQ attempts to address these questions.

Q: Is PowerDNS 2.9 really open source? What license?

A: PowerDNS 2.9 is licensed under the GNU General Public License version two, the same license that covers the Linux kernel.

Q: Is the open source version crippled?

A: It is not. Not a single byte has been omitted.

Q: Is the nameserver abandoned?

A: Far from it. In fact, we expect development to speed up now that we have joined the open source community.

Q: Why is the nameserver now open source?

A: In the current economic climate and also the way the Internet is built up right now, selling software is very hard. Most potential customers had never before bought a piece of software for their UNIX internet setup. Even though we know (from the recent survey) that nameserver operators love PowerDNS, their suggested price for it is in the $100 range.

For us, it makes far more sense to open source PowerDNS than to ask $100 for it. It is expected that open sourcing PowerDNS will lead to far higher adoption rates. We hope that PowerDNS will soon be included in major Linux and UNIX distributions.

Q: How does PowerDNS.COM BV expect to make money now that the nameserver is free?

A: In fact, we don't expect to in the near future. We also don't have a lot of expenses, basically some hosting and a few domain names.

However, we are available for consulting work, for example to help a large registrar or registry migrate to PowerDNS, or to help integrate our software in existing provisioning systems.

Furthermore, non-GPL licenses are available for those needing to do closed source modifications, or for customers uncomfortable with the GPL. This is much like what MySQL AB (http://www.mysql.com/company/index.html) is doing now.

In fact, their strategy is a lot like ours in general.

Q: Can I buy support contracts for PowerDNS?

Sure, to do so, please contact us at <sales@powerdns.com>

Q: Will you accept patches? We've added a feature

Probably - in general, it is best to discuss your intentions and needs on the <pdns-dev@mailman.powerdns.com> (subscribe (http://mailman.powerdns.com/mailman/listinfo/pdns-dev)) mailinglist before doing the work. We may have suggestions or guidelines on how you should implement the feature.

Q: PowerDNS doesn't work on my platform, will you port it?
Q: PowerDNS doesn't have feature I need, will you add it?

Be sure to ask on the <pdns-dev@mailman.powerdns.com> (subscribe (http://mailman.powerdns.com/mailman/listinfo/pdns-dev)) mailinglist. You can even hire us to do work on PowerDNS if plain asking is not persuasive enough. This might be the case if we don't currently have time for your feature, but you need it quickly anyhow, and are not in a position to submit a patch implementing it.

Q: Will PowerDNS Express (http://express.powerdns.com) be open sourced?

Perhaps, we're not yet sure.

Q: We are a Linux/Unix vendor, can we include PowerDNS?

A: Please do. In fact, we'd be very happy to work with you to make this happen. Contact <ahu@ds9a.nl> if you have specific upstream needs.

Appendix A. Backends in detail

This appendix lists several of the available backends in more detail

A.1. PipeBackend

Table A-1. PipeBackend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	No
Case	Depends
Module name	pipe
Lauch name	pipe

The PipeBackend allows for easy dynamic resolution based on a 'Coprocess' which can be written in any programming language that can read a question on standard input and answer on standard output.

Note: The Pipe Backend currently does not function under FreeBSD 4.x and 5.x, probably due to unfavorable interactions between its threading implementation and the fork system call.

Interestingly, the Linux PowerDNS binary running under the Linuxulator on FreeBSD does work.

To configure, the following settings are available:

pipe-command

Command to launch as backend. Mandatory.

pipe-timeout

Number of milliseconds to wait for an answer from the backend. If this time is ever exceeded, the backend is declared dead and a new process is spawned. Available since 2.7.

pipe-regex

If set, only questions matching this regular expression are even sent to the backend. This makes sure that most of PowerDNS does not slow down if you you reploy a slow backend. A query for the A record of 'www.powerdns.com' would be presented to the regex as 'www.powerdns.com;A'. A matching regex would be '^www.powerdns.com;.*$'.

To match only ANY and A queries for www.powerdns.com, use '^www.powerdns.com;(A|ANY)$'. Available since 2.8.

A.1.1. PipeBackend protocol

Questions come in over a file descriptor, by default standard input. Answers are sent out over another file descriptor, standard output by default.

A.1.1.1. Handshake

PowerDNS sends out 'HELO\t1', indicating that it wants to speak the protocol as defined in this document, version 1. A PowerDNS CoProcess must then send out a banner, prefixed by 'OK\t', indicating it launched successfully. If it does not support the indicated version, it should respond with FAIL, but not exit. Suggested behaviour is to try and read a further line, and wait to be terminated.

A.1.1.2. Questions

Questions come in three forms and are prefixed by a tag indicating the kind:

Q

Regular queries

AXFR

List requests, which mean that an entire zone should be listed

PING

Check if the coprocess is functioning

The question format:

```
type qname qclass qtype id ip-address
```

Fields are tab separated, and terminated with a single \n. Type is the tag above, qname is the domain the question is about. qclass is always 'IN' currently, denoting an INternet question. qtype is the kind of information desired, the record type, like A, CNAME or AAAA. id can be specified to help your

backend find an answer if the id is already known from an earlier query. You can ignore it. ip-address is the ip-address of the nameserver asking the question.

A.1.1.3. Answers

Each answer starts with a tag, possibly followed by a TAB and more data.

DATA

Indicating a successful line of DATA

END

Indicating the end of an answer - no further data

FAIL

Indicating a lookup failure. Also serves as 'END'. No further data.

LOG

For specifying things that should be logged. Can only be sent after a query and before an END line. After the tab, the message to be logged

So letting it be known that there is no data consists if sending 'END' without anything else. The answer format:

```
DATA qname qclass qtype ttl id content
```

'content' is as specified in Chapter 17. A sample dialogue may look like this:

```
Q www.ds9a.nl IN CNAME -1 213.244.168.210
DATA www.ds9a.nl IN CNAME 3600 1 ws1.ds9a.nl
Q ws1.ds9a.nl IN CNAME -1 213.244.168.210
END
Q wd1.ds9a.nl IN A -1 213.244.168.210
DATA ws1.ds9a.nl IN A 3600 1 1.2.3.4
DATA ws1.ds9a.nl IN A 3600 1 1.2.3.5
DATA ws1.ds9a.nl IN A 3600 1 1.2.3.6
END
```

This would correspond to a remote webserver 213.244.168.210 wanting to resolve the IP address of www.ds9a.nl, and PowerDNS traversing the CNAMEs to find the IP addresses of ws1.ds9a.nl Another dialogue might be:

```
Q ds9a.nl IN SOA -1 213.244.168.210
DATA ds9a.nl IN SOA 86400 1 ahu.ds9a.nl ...
END
AXFR 1
DATA ds9a.nl IN SOA 86400 1 ahu.ds9a.nl ...
DATA ds9a.nl IN NS 86400 1 ns1.ds9a.nl
DATA ds9a.nl IN NS 86400 1 ns2.ds9a.nl
DATA ns1.ds9a.nl IN A 86400 1 213.244.168.210
```

```
DATA ns2.ds9a.nl IN A 86400 1 63.123.33.135
.
.
END
```

This is a typical zone transfer.

A.1.1.4. Sample perl backend

```
#!/usr/bin/perl -w
# sample PowerDNS Coprocess backend
#

use strict;


$|=1; # no buffering

my $line=<>;
chomp($line);

unless($line eq "HELO\t1") {
print "FAIL\n";
print STDERR "Recevied '$line'\n";
<>;
exit;
}
print "OK Sample backend firing up\n"; # print our banner

while(<>)
{
print STDERR "$$ Received: $_";
chomp();
my @arr=split(/\t/);
if(@arr<6) {
print "LOG PowerDNS sent unparseable line\n";
print "FAIL\n";
next;
}

my ($type,$qname,$qclass,$qtype,$id,$ip)=split(/\t/);

if(($qtype eq "A" || $qtype eq "ANY") && $qname eq "webserver.example.com") {
print STDERR "$$ Sent A records\n";
print "DATA $qname $qclass A 3600 -1 1.2.3.4\n";
print "DATA $qname $qclass A 3600 -1 1.2.3.5\n";
print "DATA $qname $qclass A 3600 -1 1.2.3.6\n";
}
elsif(($qtype eq "CNAME" || $qtype eq "ANY") && $qname eq "www.example.com") {
print STDERR "$$ Sent CNAME records\n";
print "DATA $qname $qclass CNAME 3600 -1 webserver.example.com\n";
```

```
}
elsif($qtype eq "MBOXFW") {
print STDERR "$$ Sent MBOXFW records\n";
print "DATA $qname $qclass MBOXFW 3600 -1 powerdns\@example.com\n";
}


print STDERR "$$ End of data\n";
print "END\n";
}
```

A.2. MySQL backend

> **Warning**
>
> This backend is deprecated! Use the Generic MySQL backend which is better in *all* respects. It does support master/slave operation, this backend does not. See Section A.5.
>
> So stop reading here unless you already have a database filled with 'mysql' records.

Table A-2. MySQL backend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	Yes
Case	Insensitive
Module name	mysql
Lauch name	mysql

The MySQL Backend as present in PDNS is fixed - it requires a certain database schema to function. This schema corresponds to this create statement:

```
        CREATE TABLE records (
id int(11) NOT NULL auto_increment,
domain_id int(11) NOT NULL,
name varchar(255) NOT NULL,
type varchar(6) NOT NULL,
content varchar(255) NOT NULL,
ttl int(11) NOT NULL,
prio int(11) default NULL,
change_date int(11) default NULL,
PRIMARY KEY (id),
KEY name_index(name),
KEY nametype_index(name,type),
KEY domainid_index(domain_id)
);
```

Every domain should have a unique domain_id, which should remain identical for all records in a domain. Records with a domain_id that differs from that in the domain SOA record will not appear in a zone transfer.

The change_date may optionally be updated to the time_t (the number of seconds since midnight UTC at the start of 1970), and is in that case used to auto calculate the SOA serial number in case that is unspecified.

A.2.1. Configuration settings

WARNING! Make sure that you can actually resolve the hostname of your database without accessing the database! It is advised to supply an IP address here to prevent chicken/egg problems!

mysql-dbname

: Database name to connect to

mysql-host

: Database host to connect to

mysql-password

: Password to connect with

mysql-socket

: MySQL socket to use for connecting

mysql-table

MySQL table name. Defaults to ’records’.

mysql-user

MySQL user to connect as

A.2.2. Notes

It has been observed that InnoDB tables outperform the default MyISAM tables by a large margin. Furthermore, the default number of backends (3) should be raised to 10 or 15 for busy servers.

A.3. Random Backend

Table A-3. Random Backend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	No
Case	Depends
Module name	built in
Lauch name	random

This is a very silly backend which is discussed in Section C.1 as a demonstration on how to write a PowerDNS backend.

This backend knows about only one hostname, and only about its IP address at that. With every query, a new random IP address is generated.

It only makes sense to load the random backend in combination with a regular backend. This can be done by prepending it to the **launch=** instruction, such as **launch=random,gmysql**.

Variables:

random-hostname

Hostname for which to supply a random IP address.

A.4. MySQL PDNS backend

Table A-4. MySQL backend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	Yes
Case	Insensitive
Module name	pdns
Lauch name	pdns

This is the driver that corresponds to the set of XML-RPC tools available from PowerDNS.

The schema:

```
CREATE TABLE MailForwards (
  Id int(10) unsigned NOT NULL auto_increment,
  ZoneId int(10) unsigned NOT NULL default ’0’,
  Name varchar(255) NOT NULL default ”,
  Destination varchar(255) NOT NULL default ”,
  Flags int(11) NOT NULL default ’0’,
  ChangeDate timestamp(14) NOT NULL,
  CreateDate timestamp(14) NOT NULL,
  Active tinyint(4) NOT NULL default ’0’,
  PRIMARY KEY  (Id),
  KEY NameIndex (Name),
  KEY ZoneIdIndex (ZoneId)
);

--
-- Table structure for table ’Mailboxes’
--

CREATE TABLE Mailboxes (
  Id int(10) unsigned NOT NULL auto_increment,
  ZoneId int(10) unsigned NOT NULL default ’0’,
```

```
  Name varchar(255) NOT NULL default ”,
  Password varchar(255) NOT NULL default ”,
  Quota int(10) unsigned NOT NULL default ’0’,
  Flags int(11) NOT NULL default ’0’,
  ChangeDate timestamp(14) NOT NULL,
  CreateDate timestamp(14) NOT NULL,
  Active tinyint(4) NOT NULL default ’0’,
  PRIMARY KEY  (Id),
  UNIQUE KEY Name (Name),
  KEY ZoneIdIndex (ZoneId),
  KEY NameIndex (Name)
);

--
-- Table structure for table ’Records’
--

CREATE TABLE Records (
  Id int(10) unsigned NOT NULL auto_increment,
  ZoneId int(10) unsigned NOT NULL default ’0’,
  Name varchar(255) NOT NULL default ”,
  Type varchar(8) NOT NULL default ”,
  Content varchar(255) NOT NULL default ”,
  TimeToLive int(11) NOT NULL default ’60’,
  Priority int(11) NOT NULL default ’0’,
  Flags int(11) NOT NULL default ’0’,
  ChangeDate timestamp(14) NOT NULL,
  CreateDate timestamp(14) NOT NULL,
  Active tinyint(4) NOT NULL default ’0’,
  PRIMARY KEY  (Id),
  KEY NameIndex (Name)
);

--
-- Table structure for table ’WebForwards’
--

CREATE TABLE WebForwards (
  Id int(10) unsigned NOT NULL auto_increment,
  ZoneId int(10) unsigned NOT NULL default ’0’,
  Name varchar(255) NOT NULL default ”,
  Destination varchar(255) NOT NULL default ”,
  Type varchar(7) NOT NULL default ’NORMAL’,
  Title varchar(255) NOT NULL default ”,
  Description varchar(255) NOT NULL default ”,
  Keywords varchar(255) NOT NULL default ”,
  FavIcon varchar(255) NOT NULL default ”,
  Flags int(11) NOT NULL default ’0’,
  ChangeDate timestamp(14) NOT NULL,
  CreateDate timestamp(14) NOT NULL,
  Active tinyint(4) NOT NULL default ’0’,
  PRIMARY KEY  (Id),
  KEY NameIndex (Name),
```

```
  KEY ZoneIdIndex (ZoneId)
);

--
-- Table structure for table 'Zones'
--

CREATE TABLE Zones (
  Id int(10) unsigned NOT NULL auto_increment,
  Name varchar(255) NOT NULL default '',
  Hostmaster varchar(255) NOT NULL default '',
  Serial int(10) unsigned NOT NULL default '0',
  AutoSerial tinyint(4) NOT NULL default '0',
  Flags int(11) NOT NULL default '0',
  ChangeDate timestamp(14) NOT NULL,
  CreateDate timestamp(14) NOT NULL,
  Active tinyint(4) NOT NULL default '0',
  TimeToLive int(11) NOT NULL default '0',
  OwnerId varchar(255) NOT NULL default '',
  PRIMARY KEY  (Id),
  UNIQUE KEY Name (Name),
  KEY NameIndex (Name)
);
```

It takes a number of parameters:

pdns-dbname
: Database name to connect to

pdns-host
: Database host to connect to

pdns-password
: Password to connect with

pdns-socket
: MySQL socket to use for connecting

pdns-user
: MySQL user to connect as

A.4.1. Notes

It has been observed that InnoDB tables outperform the default MyISAM tables by a large margin. Furthermore, the default number of backends (3) should be raised to 10 or 15 for busy servers.

A.5. Generic MySQL and PgSQL backends

Table A-5. Generic PgSQL and MySQL backend capabilities

Native	Yes - but PostgreSQL does not replicate
Master	Yes
Slave	Yes
Superslave	Yes
Autoserial	NO
Case	All lower
Module name < 2.9.3	pgmysql
Module name > 2.9.2	gmysql and gpgsql
Lauch name	gmysql and gpgsql2 and gpgsql

PostgreSQL and MySQL backend with easily configurable SQL statements, allowing you to graft PDNS on any PostgreSQL or MySQL database of your choosing. Because all database schemas will be different, a generic backend is needed to cover all needs.

The template queries are expanded using the C function 'snprintf' which implies that substitutions are performed on the basis of %-place holders. To place a a % in a query which will not be substituted, use %%. Make sure to fill out the search key, often called 'name' in lower case!

There are in fact two backends, one for PostgreSQL and one for MySQL but they accept the same settings and use almost exactly the same database schema.

A.5.1. MySQL specifics

> **Warning**
>
> If using MySQL with 'slave' support enabled in PowerDNS you *must* run MySQL with a table engine that supports transactions.

In practice, great results are achieved with the 'InnoDB' tables. PowerDNS will silently function with non-transaction aware MySQLs but at one point this is going to harm your database, for example when an incoming zone transfer fails.

The default setup conforms to the following schema:

```
create table domains (
 id  INT auto_increment,
 name  VARCHAR(255) NOT NULL,
 master  VARCHAR(20) DEFAULT NULL,
 last_check  INT DEFAULT NULL,
 type  VARCHAR(6) NOT NULL,
 notified_serial INT DEFAULT NULL,
 account         VARCHAR(40) DEFAULT NULL,
 primary key (id)
)type=InnoDB;

CREATE UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
  id              INT auto_increment,
  domain_id       INT DEFAULT NULL,
  name            VARCHAR(255) DEFAULT NULL,
  type            VARCHAR(6) DEFAULT NULL,
  content         VARCHAR(255) DEFAULT NULL,
  ttl             INT DEFAULT NULL,
  prio            INT DEFAULT NULL,
  change_date     INT DEFAULT NULL,
  primary key(id)
)type=InnoDB;

CREATE INDEX rec_name_index ON records(name);
CREATE INDEX nametype_index ON records(name,type);
CREATE INDEX domain_id ON records(domain_id);

create table supermasters (
  ip VARCHAR(25) NOT NULL,
  nameserver VARCHAR(255) NOT NULL,
  account VARCHAR(40) DEFAULT NULL
```

```
);

GRANT SELECT ON supermasters TO pdns;
GRANT ALL ON domains TO pdns;
GRANT ALL ON records TO pdns;
```

This schema contains all elements needed for master, slave and superslave operation. Depending on which features will be used, the 'GRANT' statements can be trimmed to make sure PDNS cannot subvert the contents of your database.

Zone2sql with the --gmysql flag also assumes this layout is in place.

A.5.2. PostgresSQL specifics

The default setup conforms to the following schema, which you should add to a PostgreSQL database.

```
create table domains (
 id  SERIAL PRIMARY KEY,
 name  VARCHAR(255) NOT NULL,
 master  VARCHAR(20) DEFAULT NULL,
 last_check  INT DEFAULT NULL,
 type  VARCHAR(6) NOT NULL,
 notified_serial INT DEFAULT NULL,
 account         VARCHAR(40) DEFAULT NULL
);
CREATE UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
        id              SERIAL PRIMARY KEY,
        domain_id       INT DEFAULT NULL,
        name            VARCHAR(255) DEFAULT NULL,
        type            VARCHAR(6) DEFAULT NULL,
        content         VARCHAR(255) DEFAULT NULL,
        ttl             INT DEFAULT NULL,
        prio            INT DEFAULT NULL,
        change_date     INT DEFAULT NULL,
        CONSTRAINT domain_exists
        FOREIGN KEY(domain_id) REFERENCES domains(id)
        ON DELETE CASCADE
);

CREATE INDEX rec_name_index ON records(name);
CREATE INDEX nametype_index ON records(name,type);
CREATE INDEX domain_id ON records(domain_id);

create table supermasters (
```

```
  ip VARCHAR(25) NOT NULL,
  nameserver VARCHAR(255) NOT NULL,
  account VARCHAR(40) DEFAULT NULL
);

GRANT SELECT ON supermasters TO pdns;
GRANT ALL ON domains TO pdns;
GRANT ALL ON domains_id_seq TO pdns;
GRANT ALL ON records TO pdns;
GRANT ALL ON records_id_seq TO pdns;
```

This schema contains all elements needed for master, slave and superslave operation. Depending on which features will be used, the 'GRANT' statements can be trimmed to make sure PDNS cannot subvert the contents of your database.

Zone2sql with the --gpgsql flag also assumes this layout is in place.

With PostgreSQL, you may have to run 'createdb powerdns' first and then connect to that database with 'psql powerdns', and feed it the schema above.

A.5.3. Basic functionality

4 queries are needed for regular lookups, 4 for 'fancy records' which are disabled by default and 1 is needed for zone transfers.

The 4+4 regular queries must return the following 6 fields, in this exact order:

content
: This is the 'right hand side' of a DNS record. For an A record, this is the IP address for example.

ttl
: TTL of this record, in seconds. Must be a real value, no checking is performed.

prio
: For MX records, this should be the priority of the mail exchanger specified.

qtype
: The ASCII representation of the qtype of this record. Examples are 'A', 'MX', 'SOA', 'AAAA'. Make sure that this field returns an exact answer - PDNS won't recognise 'A ' as 'A'. This can be achieved by using a VARCHAR instead of a CHAR.

domain_id

Each domain must have a unique domain_id. No two domains may share a domain_id, all records in a domain should have the same. A number.

name

Actual name of a record. Must not end in a '.' and be fully qualified - it is not relative to the name of the domain!

Please note that the names of the fields are not relevant, but the order is!

As said earlier, there are 8 SQL queries for regular lookups. To configure them, set 'gmysql-basic-query' or 'gpgsql-basic-query', depending on your choice of backend. If so called 'MBOXFW' fancy records are not used, four queries remain:

basic-query

Default: **select content,ttl,prio,type,domain_id,name from records where qtype='%s' and name='%s'** This is the most used query, needed for doing 1:1 lookups of qtype/name values. First %s is replaced by the ASCII representation of the qtype of the question, the second by the name.

id-query

Default: **select content,ttl,prio,type,domain_id,name from records where qtype='%s' and name='%s' and domain_id=%d** Used for doing lookups within a domain. First %s is replaced by the qtype, the %d which should appear after the %s by the numeric domain_id.

any-query

For doing ANY queries. Also used internally. Default: **select content,ttl,prio,type,domain_id,name from records where name='%s'** The %s is replaced by the qname of the question.

any-id-query

For doing ANY queries within a domain. Also used internally. Default: **select content,ttl,prio,type,domain_id,name from records where name='%s' and domain_id=%d** The %s is replaced by the name of the domain, the %d by the numerical domain id.

The last query is for listing the entire contents of a zone. This is needed when performing a zone transfer, but sometimes also internally:

list-query

To list an entire zone. Default: **select content,ttl,prio,type,domain_id,name from records where domain_id=%d**

A.5.4. Master/slave queries

Most installations will have zero need to change the following settings, but should the need arise, here they are:

master-zone-query

Called to determine the master of a zone. Default: **select master from domains where name='%s' and type='SLAVE'**

info-zone-query

Called to retrieve (nearly) all information for a domain: Default: **select id,name,master,last_check,notified_serial,type from domains where name='%s'**

info-all-slaves-query

Called to retrieve all slave domains Default: **select id,name,master,last_check,type from domains where type='SLAVE'**

supermaster-query

Called to determine if a certain host is a supermaster for a certain domain name. Default: **select account from supermasters where ip='%s' and nameserver='%s'");**

insert-slave-query

Called to add a domain as slave after a supermaster notification. Default: **insert into domains (type,name,master,account) values('SLAVE','%s','%s','%s')**

insert-record-query

Called during incoming AXFR. Default: **insert into records (content,ttl,prio,type,domain_id,name) values ('%s',%d,%d,'%s',%d,'%s')**

update-serial-query

Called to update the last notified serial of a master domain. Default: **update domains set notified_serial=%d where id=%d**

update-lastcheck-query

Called to update the last time a slave domain was checked for freshness. Default: **update domains set notified_serial=%d where id=%d**

info-all-master-query

Called to get data on all domains for which the server is master. Default: **select id,name,master,last_check,notified_serial,type from domains where type='MASTER'**

delete-zone-query

Called to delete all records of a zone. Used before an incoming AXFR. Default: **delete from records where domain_id=%d**

A.5.5. Fancy records

If PDNS is used with so called 'Fancy Records', the 'MBOXFW' record exists which specifies an email address forwarding instruction, wildcard queries are sometimes needed. This is not enabled by default. A wildcard query is an internal concept - it has no relation to *.domain-type lookups. You can safely leave these queries blank.

wildcard-query

Can be left blank. See above for an explanation. Default: **select content,ttl,prio,type,domain_id,name from records where qtype='%s' and name like '%s'**

wildcard-id-query

Can be left blank. See above for an explanation. Default: **select content,ttl,prio,type,domain_id,name from records where qtype='%s' and name like '%s' and domain_id=%d** Used for doing lookups within a domain.

wildcard-any-query

For doing wildcard ANY queries. Default: **select content,ttl,prio,type,domain_id,name from records where name like '%s'**

wildcard-any-id-query

For doing wildcard ANY queries within a domain. Default: **select content,ttl,prio,type,domain_id,name from records where name like '%s' and domain_id=%d**

A.5.6. Settings and specifying queries

The queries above are specified in pdns.conf. For example, the basic-query would appear as:

```
gpgsql-basic-query=select content,ttl,prio,type,domain_id,name from records where qtype
```

When using the Generic PostgreSQL backend, they appear as above. When using the generic MySQL backend, change the "gpgsql-" prefix to "gmysql-".

Queries can span multiple lines, like this:

```
gpgsql-basic-query=select content,ttl,prio,type,domain_id,name from records \
where qtype='%s' and name='%s'
```

Do not wrap statements in quotes as this will not work. Besides the query related settings, the following configuration options are available:

gpgsql-dbname

Database name to connect to

gpgsql-host

Database host to connect to. WARNING: When specified as a hostname a chicken/egg situation might arise where the database is needed to resolve the IP address of the database. It is best to supply an IP address of the database here.

gmysql-host (only for MySQL!)

Filename where the MySQL connection socket resides. Often `/tmp/mysql.sock` or `/var/run/mysqld/mysqld.sock`.

gpgsql-password

Password to connect with

gpgsql-user

PgSQL user to connect as

A.5.7. Native operation

For native operation, either drop the FOREIGN KEY on the domain_id field, or (recommended), make sure the **domains** table is filled properly. To add a domain, issue the following:

```
insert into domains (name,type) values ('powerdns.com','NATIVE');
```

The records table can now be filled by with the domain_id set to the id of the domains table row just inserted.

A.5.8. Slave operation

The PostgreSQL backend is fully slave capable. To become a slave of the 'powerdns.com' domain, execute this:

```
insert into domains (name,master,type) values ('powerdns.com','213.244.168.217','SLAVE'
```

And wait a while for PDNS to pick up the addition - which happens within one minute. There is no need to inform PDNS that a new domain was added. Typical output is:

```
Apr 09 13:34:29 All slave domains are fresh
Apr 09 13:35:29 1 slave domain needs checking
Apr 09 13:35:29 Domain powerdns.com is stale, master serial 1, our serial 0
Apr 09 13:35:30 [gPgSQLBackend] Connected to database
Apr 09 13:35:30 AXFR started for 'powerdns.com'
```

```
Apr 09 13:35:30 AXFR done for 'powerdns.com'
Apr 09 13:35:30 [gPgSQLBackend] Closing connection
```

From now on, PDNS is authoritative for the 'powerdns.com' zone and will respond accordingly for queries within that zone.

Periodically, PDNS schedules checks to see if domains are still fresh. The default **slave-cycle-interval** is 60 seconds, large installations may need to raise this value. Once a domain has been checked, it will not be checked before its SOA refresh timer has expired. Domains whose status is unknown get checked every 60 seconds by default.

A.5.9. Superslave operation

To configure a supermaster with IP address 10.0.0.11 which lists this installation as 'autoslave.powerdns.com', issue the following:

```
insert into supermasters ('10.0.0.11','autoslave.powerdns.com','internal');
```

From now on, valid notifies from 10.0.0.11 that list a NS record containing 'autoslave.powerdns.com' will lead to the provisioning of a slave domain under the account 'internal'. See Section 13.2.1 for details.

A.5.10. Master operation

The PostgreSQL backend is fully master capable with automatic discovery of serial changes. Raising the serial number of a domain suffices to trigger PDNS to send out notifications. To configure a domain for master operation instead of the default native replication, issue:

```
insert into domains (name,type) values ('powerdns.com','MASTER');
```

Make sure that the assigned id in the domains table matches the domain_id field in the records table!

A.6. Generic Oracle backend

Table A-6. Oracle backend capabilities

Native	Yes
Master	No

Slave	No
Superslave	No
Autoserial	Yes
Module name	oracle
Launch name	oracle

Oracle backend with easily configurable SQL statements, allowing you to graft PDNS on any Oracle database of your choosing.

PowerDNS is currently ascertaining if this backend can be distributed in binary form without violating Oracle licensing. In the meantime, the source code to the Oracle backend is available in the pdns distribution.

The following configuration settings are available:

oracle-debug-queries

Output all queries to disk for debugging purposes.

oracle-time-queries

Output all queries to disk for timing purposes.

oracle-uppercase-database

Change all domain names to uppercase before querying database.

oracle-database

Oracle database name to connect to.

oracle-home

PDNS can set the ORACLE_HOME environment variable from within the executable, allowing execution of the daemon from init.d scripts where ORACLE_HOME may not yet be set.

oracle-sid

PDNS can set the ORACLE_SID environment variable from within the executable, allowing execution of the daemon from init.d scripts where ORACLE_SID may not yet be set.

oracle-username

Oracle username to connect as.

oracle-password

Oracle password to connect with.

The generic Oracle backend can be configured to use user-specified queries. The following are the default queries and their names:

oracle-forward-query

select content, TimeToLive, Priority, type, ZoneId, nvl(ChangeDate,0) from Records where name = :name and type = :type

oracle-forward-query-by-zone

select content, TimeToLive, Priority, type, ZoneId, nvl(ChangeDate,0) from records where name = :name and type = :type and ZoneId = :id

oracle-forward-any-query

select content, TimeToLive, Priority, type, ZoneId, nvl(ChangeDate,0) from records where name = :name

oracle-list-query

select content, TimeToLive, Priority, type, ZoneId, nvl(ChangeDate, 0), name from records where ZoneId = :id

A.6.1. Setting up Oracle for use with PowerDNS

To setup a database that corresponds to these default queries, issue the following as Oracle user sys:

```
create user powerdns identified by YOURPASSWORD;
grant connect to powerdns;

create tablespace powerdns datafile '/opt/oracle/oradata/oracle/powerdns.dbf'
  size 256M extent management local autoallocate;

alter user powerdns quota unlimited on powerdns;
```

As user 'powerdns' continue with:

```
create table Domains (
 ID  number(11) NOT NULL,
 NAME  VARCHAR(255) NOT NULL,
```

```
 MASTER  VARCHAR(20) DEFAULT NULL,
 LAST_CHECK  INT DEFAULT NULL,
 TYPE  VARCHAR(6) NOT NULL,
 NOTIFIED_SERIAL INT DEFAULT NULL,
 ACCOUNT  VARCHAR(40) DEFAULT NULL,
 primary key (ID)
)tablespace POWERDNS;

create index DOMAINS$NAME on Domains (NAME) tablespace POWERDNS;
create sequence DOMAINS_ID_SEQUENCE;

create table Records
(
  ID               number(11) NOT NULL,
  ZoneID           number(11) default NULL REFERENCES Domains(ID) ON DELETE CASCADE,
  NAME             varchar2(255) default NULL,
  TYPE             varchar2(6) default NULL,
  CONTENT          varchar2(255) default NULL,
  TimeToLive       number(11) default NULL,
  Priority         number(11) default NULL,
  CreateDate       number(11) default NULL,
  ChangeDate       number(11) default NULL,
  primary key (ID)
)tablespace POWERDNS;

create index RECORDS$NAME on RECORDS (NAME) tablespace POWERDNS;
create sequence RECORDS_ID_SEQUENCE;
```

To insert records, either use **zone2sql** with the **--oracle** setting, or execute sql along the lines of:

```
insert into domains (id,name,type) values (domains_id_sequence.nextval,'netherlabs.nl','NAT
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
insert into Records (id,ZoneId, name,type,content,TimeToLive,Priority) select RECORDS_ID_SE
```

For performance reasons it is best to specify **--transactions** too!

A.7. DB2 backend

Table A-7. DB2 backend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	Yes
Module name	db2
Launch name	db2

PowerDNS is currently ascertaining if this backend can be distributed in binary form without violating IBM DB2 licensing.

The DB2 backend executes the following queries:

Forward Query

select Content, TimeToLive, Priority, Type, ZoneId, 0 as ChangeDate, Name from Records where Name = ? and type = ?

Forward By Zone Query

select Content, TimeToLive, Priority, Type, ZoneId, 0 as ChangeDate, Name from Records where Name = ? and Type = ? and ZoneId = ?

Forward Any Query

select Content, TimeToLive, Priority, Type, ZoneId, 0 as ChangeDate, Name from Records where Name = ?

List Query

select Content, TimeToLive, Priority, Type, ZoneId, 0 as ChangeDate, Name from Records where ZoneId = ?

Configuration settings:

db2-server

Server name to connect to. Defaults to 'powerdns'. Make sure that your nameserver is not needed to resolve an IP address needed to connect as this might lead to a chicken/egg situation.

db2-user

Username to connect as. Defaults to ’powerdns’.

db2-password

Password to connect with. Defaults to ’powerdns’.

A.8. Bind zone file backend

Table A-8. Bind zone file backend capabilities

Native	Yes
Master	Yes
Slave	Yes
Superslave	No
Autoserial	No
Module name	none (built in)
Launch	bind

The BindBackend started life as a demonstration of the versatility of PDNS but quickly gained in importance when there appeared to be demand for a Bind ’workalike’.

The BindBackend parses a Bind-style named.conf and extracts information about zones from it. It makes no attempt to honour other configuration flags, which you should configure (when available) using the PDNS native configuration.

--help=bind

Outputs all known parameters related to the bindbackend

bind-example-zones

Loads the ’example.com’ zone which can be queried to determine if PowerDNS is functioning without configuring database backends.

bind-config=

Location of the Bind configuration file to parse.

bind-check-interval=

How often to check for zone changes. See ’Operation’ section.

bind-enable-huffman

Enable Huffman compression on zone data. Currently saves around 20% of memory actually used, but slows down operation somewhat.

A.8.1. Operation

On launch, the BindBackend first parses the named.conf to determine which zones need to be loaded. These will then be parsed and made available for serving, as they are parsed. So a named.conf with 100.000 zones may take 20 seconds to load, but after 10 seconds, 50.000 zones will already be available. While a domain is being loaded, it is not yet available, to prevent incomplete answers.

Reloading is currently done only when a request for a zone comes in, and then only after **bind-check-interval** seconds have passed after the last check. If a change occurred, access to the zone is disabled, the file is reloaded, access is restored, and the question is answered. For regular zones, reloading is fast enough to answer the question which lead to the reload within the DNS timeout.

If **bind-check-interval** is specified as zero, no checks will be performed until the **pdns_control reload** is given.

A.8.2. Pdns_control commands

bind-domain-status **`domain`** [**`domain`**]

Output status of domain or domains. Can be one of ’seen in named.conf, not parsed’, ’parsed successfully at <time;>’ or ’error parsing at line ... at <time>’.

bind-list-rejects

Lists all zones that have problems, and what those problems are.

bind-reload-now **`domain`**

Reloads a zone from disk NOW, reporting back results.

A.8.3. Performance

The BindBackend does not benefit from the packet cache as it is fast enough on its own. Furthermore, on most systems, there will be no benefit in using multiple CPUs for the packetcache, so a noticeable

speedup can be attained by specifying **distributor-threads=1** in `pdns.conf`.

A.8.4. Master/slave configuration

A.8.4.1. Master

Works as expected. At startup, no notification storm is performed as this is generally not useful. Perhaps in the future the Bind Backend will attempt to store zone metadata in the zone, allowing it to determine if a zone has changed its serial since the last time notifications were sent out.

Changes which are discovered when reloading zones do lead to notifications however.

A.8.4.2. Slave

Also works as expected. The Bind backend expects to be able to write to a directory where a slave domain lives. The incoming zone is stored as 'zonename.RANDOM' and atomically renamed if it is retrieved successfully, and parsed only then.

In the future, this may be improved so the old zone remains available should parsing fail.

A.8.5. Commands

pdns_control offers commands to communicate instructions to PowerDNS. These are detailed here.

rediscover

Reread the bind configuration file (`named.conf`). If parsing fails, the old configuration remains in force and pdns_control reports the error. Any newly discovered domains are read, discarded domains are removed from memory.

Note: Except that with 2.9.3, they are not removed from memory.

bind-reload

All zones with a changed timestamp are reloaded at the next incoming query for them.

A.9. ODBC backend

Table A-9. ODBC backend capabilities

Native	Yes
Master	Yes (experimental)
Slave	Yes (experimental)
Superslave	No
Autoserial	Yes

The ODBC backend can retrieve zone information from any source that has a ODBC driver available.

Note: This backend is only available on PowerDNS for Windows.

The ODBC backend needs data in a fixed schema which is the same as the data needed by the MySQL backend. The create statement will resemble this:

```
        CREATE TABLE records (
id int(11) NOT NULL auto_increment,
domain_id int(11) default NULL,
name varchar(255) default NULL,
type varchar(6) default NULL,
content varchar(255) default NULL,
ttl int(11) default NULL,
prio int(11) default NULL,
change_date int(11) default NULL,
PRIMARY KEY (id),
KEY name_index(name),
KEY nametype_index(name,type),
KEY domainid_index(domain_id)
);
```

To use the ODBC backend an ODBC source has to be created, to do this see the section Installing PowerDNS on Microsoft Windows, Chapter 3.

The following configuration settings are available:

odbc-datasource

Specifies the name of the data source to use.

odbc-user

Specifies the username that has to be used to log into the datasource.

odbc-pass

Specifies the user's password.

odbc-table

Specifies the name of the table containing the zone information.

The ODBC backend has been tested with Microsoft Access, MySQL (via MyODBC) and Microsoft SQLServer. As the SQL statements used are very basic, it is expected to work with many ODBC drivers.

A.10. XDB Backend

Special purpose backend for grandiose performance. Can talk to Tridge's Trivial Database, or to regular *db tables on disk. Currently only sparsely documented. Very useful if you need to do >50.000 queries/second, which we actually measured on the .ORG zone.

More documentation will follow.

A.11. LDAP backend

The main author for this module is Norbert Sendetzky who also has his own PowerDNS-LDAP page (http://www.linuxnetworks.de/pdnsldap/index.html).

Table A-10. LDAP backend capabilities

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	Yes

As of 2.9.6, PowerDNS comes with an LDAP backend. The code for this was submitted by Norbert Sendetzky.

The following settings are available to configure the LDAP backend:

ldap-host

LDAP host to connect to, defaults to localhost.

ldap-port

LDAP port to connect to, defaults to 389.

ldap-basedn

Root for DNS searches. Must be configured before the LDAP backend will work.

ldap-binddn

Distinguished Name to bind with to the LDAP server. Defaults to the empty string for anonymous bind.

ldap-secret

Secret to bind with to LDAP server. Defaults to the empty string for anonymous bind.

ldap-default-ttl

TTL for records with no dnsttl attribute. Defaults to 86400 seconds.

The schema used is that defined by RFC 1279 and is present in OpenLDAP under the name 'cosine.schema'. An example LDIF file:

```
# zone related things including SOA, NS and MX records

dn: dc=example
objectclass: top
objectclass: dnsdomain
objectclass: domainrelatedobject
dc: example
soarecord: ns.example.dom hostmaster@example.dom 2002010401 1800 3600 604800 84600
nsrecord: ns.example.dom
mxrecord: 10 mail.example.dom
mxrecord: 20 mail2.example.dom
associateddomain: example.dom


# Simple record (mail.example.dom has address 172.168.0.2)

dn: dc=mail,dc=example
objectclass: top
objectclass: dnsdomain
```

```
objectclass: domainrelatedobject
dc: mail
arecord: 172.168.0.2
associateddomain: mail.example.dom

# There may more than one entry per record
# This is also applicable to all other records including "associateddomain"
# but not for a CNAME record

dn: dc=server,dc=snapcount
objectclass: top
objectclass: dnsdomain
objectclass: domainrelatedobject
dc: server
arecord: 10.1.0.1
arecord: 172.168.0.1
associateddomain: server.example.dom


# domain alias ({mail2,ns}.example.dom is CNAME for server.example.dom)
# cnamerecord must only contain one entry

dn: dc=backup,dc=snapcount
objectclass: top
objectclass: dnsdomain
objectclass: domainrelatedobject
dc: server
cnamerecord: server.example.dom
associateddomain: mail2.example.dom
associateddomain: ns.example.dom
```

Appendix B. PDNS internals

PDNS is normally launched by the init.d script but is actually a binary called `pdns_server`. This file is started by the **start** and **monitor** commands to the init.d script. Other commands are implemented using the controlsocket.

B.1. Controlsocket

The controlsocket is the means to contact a running PDNS daemon, or as we now know, a running `pdns_server`. Over this sockets, instructions can be sent using the `pdns_control` program. Like the `pdns_server`, this program is normally accessed via the init.d script.

B.1.1. pdns_control

To communicate with PDNS over the controlsocket, the **pdns_control** command is used. The init.d script also calls pdns_control. The syntax is simple: **pdns_control command arguments**. Currently this is most useful for telling backends to rediscover domains or to force the transmission of notifications. See Section 13.3.

Besides the commands implemented by the init.d script, for which see Section 2.3, the following pdns_control commands are available:

ccounts

Returns counts on the contents of the cache.

notify **`domain`**

Adds a domain to the notification list, causing PDNS to send out notifications to the nameservers of a domain. Can be used if a slave missed previous notifications or is generally hard of hearing.

notify-host **`domain host`**

Same as above but with operator specified IP address as destination, to be used if you know better than PowerDNS.

purge

Purges the entire Packet Cache - see Chapter 9.

purge **`record`**

Purges all entries for this exact record name - see Chapter 9.

purge **`record`**$

Purges all cache entries ending on this name, effectively purging an entire domain - see Chapter 9.

purge

Purges the entire Packet Cache - see Chapter 9.

purge **`record`**

Purges all entries for this exact record name - see Chapter 9.

rediscover

Instructs backends that new domains may have appeared in the database, or, in the case of the Bind backend, in named.conf.

reload

Instructs backends that the contents of domains may have changed. Many backends ignore this, the Bind backend will check timestamps for all zones (once queries come in for it) and reload if needed.

retrieve **`domain`**

Retrieve a slave domain from its master. Done nearly immediatly.

set **`variable value`**

Set a configuration parameter. Currently only the 'query-logging' parameter can be set.

uptime

Reports the uptime of the daemon in human readable form.

version

returns the version of a running pdns daemon.

B.2. Guardian

When launched by the init.d script, `pdns_server` wraps itself inside a 'guardian'. This guardian monitors the performance of the inner `pdns_server` instance which shows up in the process list of your OS as `pdns_server-instance`. It is also this guardian that `pdns_control` talks to. A **STOP** is interpreted by the guardian, which causes the guardian to sever the connection to the inner process and terminate it, after which it terminates itself. The init.d script **DUMP** and **SHOW** commands need to access the inner process, because the guardian itself does not run a nameserver. For this purpose, the guardian passes controlsocket requests to the control console of the inner process. This is the same console as seen with init.d **MONITOR**.

B.3. Modules & Backends

PDNS has the concept of backends and modules. Non-static PDNS distributions have the ability to load new modules at runtime, while the static versions come with a number of modules built in, but cannot

load more.

Related parameters are:

--help

Outputs all known parameters, including those of launched backends, see below.

--launch=backend,backend1,backend1:name

Launches backends. In its most simple form, supply all backends that need to be launched. If you find that you need to launch single backends multiple times, you can specify a name for later instantiations. In this case, there are 2 instances of backend1, and the second one is called 'name'. This means that **--backend1-setting** is available to configure the first or main instance, and **--backend1-name-setting** for the second one.

--load-modules=/directory/libyourbackend.so

If backends are available in nonstandard directories, specify their location here. Multiple files can be loaded if separated by commas. Only available in non-static PDNS distributions.

--list-modules

Will list all available modules, both compiled in and in dynamically loadable modules.

To run on the commandline, use the **pdns_server** binary. For example, to see options for the gpgsql backend, use the following:

```
$ /usr/sbin/pdns_server --launch=gpgsql --help=gpgsql
```

B.4. How PDNS translates DNS queries into backend queries

A DNS query is not a straightforward lookup. Many DNS queries need to check the backend for additional data, for example to determine of an unfound record should lead to an NXDOMAIN ('we know about this domain, but that record does not exist') or an unauthoritative response.

Simplified, without CNAME processing and wildcards, the algorithm is like this:

When a query for a **qname**/**qtype** tuple comes in, it is requested directly from the backend. If present, PDNS adds the contents of the reply to the list of records to return. A question tuple may generate multiple answer records.

Each of these records is now investigated to see if it needs 'additional processing'. This holds for example for MX records which may point to hosts for which the PDNS backends also contain data. This involves further lookups for A or AAAA records.

After all additional processing has been performed, PDNS sieves out all double records which may well have appeared. The resulting set of records is added to the answer packet, and sent out.

A zone transfer works by looking up the **domain_id** of the SOA record of the name and then listing all records of that **domain_id**. This is why all records in a domain need to have the same domain_id.

When a query comes in for an unknown domain, PDNS starts looking for SOA records of all subdomains of the qname, so no.such.powerdns.com turns into a SOA query for no.such.powerdns.com, such.powerdns.com, powerdns.com, com, ". When a SOA is found, that zone is consulted for relevant NS instructions which lead to a referral. If nothing is found within the zone, an authoritative NXDOMAIN is sent out.

If no SOA was found, an unauthoritative no-error is returned.

In reality, each query for a question tuple first involves checking for a CNAME, unless that resolution has been disabled with the **skip-cname** option.

PDNS breaks strict RFC compatability by not always checking for the presence of a SOA record first. This is unlikely to lead to problems though.

Appendix C. Backend writers' guide

PDNS backends are implemented via a simple yet powerful C++ interface. If your needs are not met by the PipeBackend, you may want to write your own. Doing so requires a copy of the PowerDNS Open Source Backend Development kit which can be found on http://downloads.powerdns.com/releases/dev.

A backend contains zero DNS logic. It need not look for CNAMES, it need not return NS records unless explicitly asked for, etcetera. All DNS logic is contained within PDNS itself - backends should simply return records matching the description asked for.

> **Warning**
>
> However, please note that your backend can get queries in aNy CAsE! If your database is case sensitive, like most are (with the notable exception of MySQL), you must make sure that you do find answers which differ only in case.

C.1. Simple read-only native backends

Implementing a backend consists of inheriting from the DNSBackend class. For read-only backends, which do not support slave operation, only the following methods are relevant:

```
class DNSBackend
{
public:

virtual bool lookup(const QType &qtype, const string &qdomain, DNSPacket *pkt_p=0, int zone
virtual bool list(int domain_id)=0;
virtual bool get(DNSResourceRecord &r)=0;
virtual bool getSOA(const string &name, SOAData &soadata);
};
```

Note that the first three methods must be implemented. `getSOA()` has a useful default implementation.

The semantics are simple. Each instance of your class only handles one (1) query at a time. There is no need for locking as PDNS guarantees that your backend will never be called reentrantly.

Some examples, a more formal specification is down below. A normal lookup starts like this:

```
        YourBackend yb;
yb.lookup(QType::CNAME,"www.powerdns.com");
```

Your class should now do everything to start this query. Perform as much preparation as possible - handling errors at this stage is better for PDNS than doing so later on. A real error should be reported by throwing an exception.

PDNS will then call the `get()` method to get **DNSResourceRecord**s back. The following code illustrates a typical query:

```
yb.lookup(QType::CNAME,"www.powerdns.com");

DNSResourceRecord rr;
while(yb.get(rr))
      cout<<"Found cname pointing to '"+rr.content+"'"<<endl;
}
```

Each zone starts with a Start of Authority (SOA) record. This record is special so many backends will choose to implement it specially. The default `getSOA()` method performs a regular lookup on your backend to figure out the SOA, so if you have no special treatment for SOA records, where is no need to implement your own `getSOA()`.

Besides direct queries, PDNS also needs to be able to list a zone, to do zone transfers for example. Each zone has an id which should be unique within the backend. To list all records belonging to a zone id, the `list()` method is used. Conveniently, the domain_id is also available in the **SOAData** structure.

The following lists the contents of a zone called "powerdns.com".

```
SOAData sd;
if(!yb.getSOA("powerdns.com",sd))  // are we authoritative over powerdns.com?
  return RCode::NotAuth;           // no

yb.list(sd.domain_id);
while(yb.get(rr))
      cout<<rr.qname<<"\t IN "<<rr.qtype.getName()<<"\t"<<rr.content<<endl;
```

Please note that when so called 'fancy records' (see Chapter 14) are enabled, a backend can receive wildcard lookups. These have a % as the first character of the qdomain in lookup.

C.1.1. A sample minimal backend

This backend only knows about the host "random.powerdns.com", and furthermore, only about its A record:

```
/* FIRST PART */
```

```
class RandomBackend : public DNSBackend
{
public:
  bool list(int id) {
    return false; // we don't support AXFR
  }

  void lookup(const QType &type, const string &qdomain, DNSPacket *p, int zoneId)
  {
    if(type.getCode()!=QType::A || qdomain!="random.powerdns.com")  // we only know about r
      d_answer="";                                                  // no answer
    else {
      ostringstream os;
      os<<random()%256<<"."<<random()%256<<"."<<random()%256<<"."<<random()%256;
      d_answer=os.str();                                           // our random ip address
    }
  }

  bool get(DNSResourceRecord &rr)
  {
    if(!d_answer.empty()) {
      rr.qname="random.powerdns.com";                               // fill in details
      rr.qtype=QType::A;                                            // A record
      rr.ttl=86400;                                                 // 1 day
      rr.content=d_answer;

      d_answer="";                                                  // this was the last an

      return true;
    }
    return false;                                                   // no more data
  }

private:
  string d_answer;
};

/* SECOND PART */

class RandomFactory : public BackendFactory
{
public:
  RandomFactory() : BackendFactory("random") {}

  DNSBackend *make(const string &suffix)
  {
    return new RandomBackend();
  }
};

/* THIRD PART */

class RandomLoader
```

```
{
public:
  Loader()
  {
    BackendMakers().report(new RandomFactory);

    L<<Logger::Info<<" [RandomBackend] This is the randombackend ("__DATE__", "__TIME__")
  }
};

static RandomLoader randomloader;
```

This simple backend can be used as an 'overlay'. In other words, it only knows about a single record, another loaded backend would have to know about the SOA and NS records and such. But nothing prevents us from loading it without another backend.

The first part of the code contains the actual logic and should be pretty straightforward. The second part is a boilerplate 'factory' class which PDNS calls to create randombackend instances. Note that a 'suffix' parameter is passed. Real life backends also declare parameters for the configuration file; these get the 'suffix' appended to them. Note that the "random" in the constructor denotes the name by which the backend will be known.

The third part registers the RandomFactory with PDNS. This is a simple C++ trick which makes sure that this function is called on execution of the binary or when loading the dynamic module.

Please note that a RandomBackend is actually in most PDNS releases. By default it lives on random.example.com, but you can change that by setting **random-hostname**.

NOTE: this simple backend neglects to handle case properly! For a more complete example, see the full pdns-dev distribution as found on the website (http://www.powerdns.com/pdns).

C.1.2. Interface definition

Classes:

Table C-1. DNSResourceRecord class

QType qtype	QType of this record
string qname	name of this record
string content	ASCII representation of right hand side
u_int16_t priority	priority of an MX record.
u_int32_t ttl	Time To Live of this record
int domain_id	ID of the domain this record belongs to
time_t last_modified	If unzero, last time_t this record was changed

Table C-2. SOAData struct

string nameserver	Name of the master nameserver of this zone
string hostmaster	Hostmaster of this domain. May contain an @
u_int32_t serial	Serial number of this zone
u_int32_t refresh	How often this zone should be refreshed
u_int32_t retry	How often a failed zone pull should be retried.
u_int32_t expire	If zone pulls failed for this long, retire records
u_int32_t default_ttl	Difficult
int domain_id	The ID of the domain within this backend. Must be filled!
DNSBackend *db	Pointer to the backend that feels authoritative for a domain and can act as a slave

Methods:

void lookup(const QType &qtype, const string &qdomain, DNSPacket *pkt=0, int zoneId=-1)

This function is used to initiate a straight lookup for a record of name 'qdomain' and type 'qtype'. A QType can be converted into an integer by invoking its `getCode()` method and into a string with the `getCode()`.

The original question may or may not be passed in the pointer p. If it is, you can retrieve (from 1.99.11 onwards) information about who asked the question with the `getRemote(DNSPacket *)` method. Alternatively, `bool getRemote(struct sockaddr *sa, socklen_t *len)` is available.

Note that **qdomain** can be of any case and that your backend should make sure it is in effect case insensitive. Furthermore, the case of the original question should be retained in answers returned by `get()`!

Finally, the domain_id might also be passed indicating that only answers from the indicated zone need apply. This can both be used as a restriction or as a possible speedup, hinting your backend where the answer might be found.

If initiated succesfully, as indicated by returning **true**, answers should be made available over the `get()` method.

Should throw an AhuException if an error occured accessing the database. Returning otherwise indicates that the query was started succesfully. If it is known that no data is available, no exception should be thrown! An exception indicates that the backend considers itself broken - not that no answers are available for a question.

It is legal to return here, and have the first call to `get()` return false. This is interpreted as 'no data'

bool list(int domain_id)

Initiates a list of the indicated domain. Records should then be made available via the `get()` method. Need not include the SOA record. If it is, PDNS will not get confused.

Should return false if the backend does not consider itself authoritative for this zone. Should throw an AhuException if an error occured accessing the database. Returning true indicates that data is or should be available.

bool get(DNSResourceRecord &rr)

Request a DNSResourceRecord from a query started by `get()` of `list()`. If this functions returns **true**, **rr** has been filled with data. When it returns false, no more data is available, and **rr** does not contain new data. A backend should make sure that it either fills out all fields of the DNSResourceRecord or resets them to their default values.

The qname field of the DNSResourceRecord should be filled out with the exact `qdomain` passed to lookup, preserving its case. So if a query for 'CaSe.yourdomain.com' comes in and your database contains dat afor 'case.yourdomain.com', the qname field of rr should contin 'CaSe.yourdomain.com'!

Should throw an AhuException in case a database error occurred.

bool getSOA(const string &name, SOAData &soadata)

If the backend considers itself authoritative over domain `name`, this method should fill out the passed **SOAData** structure and return a positive number. If the backend is functioning correctly, but does not consider itself authoritative, it should return 0. In case of errors, an AhuException should be thrown.

C.2. Reporting errors

To report errors, the Logger class is available which works mostly like an iostream. Example usage is as shown above in the RandomBackend. Note that it is very important that each line is ended with **endl** as

your message won't be visible otherwise.

To indicate the importance of an error, the standard syslog errorlevels are available. They can be set by outputting `Logger::Critical`, `Logger::Error`, `Logger::Warning`, `Logger::Notice`, `Logger::Info` or `Logger::Debug` to `L`, in descending order of graveness.

C.3. Declaring and reading configuration details

It is highly likely that a backend needs configuration details. On launch, these parameters need to be declared with PDNS so it knows it should accept them in the configuration file and on the commandline. Furthermore, they will be listed in the output of **--help**.

Declaring arguments is done by implementing the member function `declareArguments()` in the factory class of your backend. PDNS will call this method after launching the backend.

In the `declareArguments()` method, the function `declare()` is available. The exact definitions:

void declareArguments(const string &suffix="")

This method is called to allow a backend to register configurable parameters. The suffix is the sub-name of this module. There is no need to touch this suffix, just pass it on to the declare method.

void declare(const string &suffix, const string &param, const string &explanation, const string &value)

The suffix is passed to your method, and can be passed on to declare. **param** is the name of your parameter. **explanation** is what will appear in the output of --help. Furthermore, a default value can be supplied in the **value** parameter.

A sample implementation:

```
void declareArguments(const string &suffix)
{
    declare(suffix,"dbname","Pdns backend database name to connect to","powerdns");
  declare(suffix,"user","Pdns backend user to connect as","powerdns");
  declare(suffix,"host","Pdns backend host to connect to","");
  declare(suffix,"password","Pdns backend password to connect with","");
}
```

After the arguments have been declared, they can be accessed from your backend using the `mustDo()`, `getArg()` and `getArgAsNum()` methods. The are defined as follows in the DNSBackend class:

void setArgPrefix(const string &prefix)

Must be called before any of the other accessing functions are used. Typical usage is `'setArgPrefix("mybackend"+suffix)'` in the constructor of a backend.

bool mustDo(const string &key)

Returns true if the variable `key` is set to anything but 'no'.

const string& getArg(const string &key)

Returns the exact value of a parameter.

int getArgAsNum(const string &key)

Returns the numerical value of a parameter. Uses `atoi()` internally

Sample usage from the BindBackend, using the **bind-example-zones** and **bind-config** parameters.

```
if(mustDo("example-zones")) {
  insert(0,"www.example.com","A","1.2.3.4");
  /* ... */
}


if(!getArg("config").empty()) {
  BindParser BP;

  BP.parse(getArg("config"));
}
```

C.4. Read/write slave-capable backends

The backends above are 'natively capable' in that they contain all data relevant for a domain and do not pull in data from other nameservers. To enable storage of information, a backend must be able to do more.

Before diving into the details of the implementation some theory is in order. Slave domains are pulled from the master. PDNS needs to know for which domains it is to be a slave, and for each slave domain, what the IP address of the master is.

A slave zone is pulled from a master, after which it is 'fresh', but this is only temporary. In the SOA record of a zone there is a field which specifies the 'refresh' interval. After that interval has elapsed, the

slave nameserver needs to check at the master ff the serial number there is higher than what is stored in the backend locally.

If this is the case, PDNS dubs the domain 'stale', and schedules a transfer of data from the remote. This transfer remains scheduled until the serial numbers remote and locally are identical again.

This theory is implemented by the `getUnfreshSlaveInfos` method, which is called on all backends periodically. This method fills a vector of **SlaveDomain**s with domains that are unfresh and possibly stale.

PDNS then retrieves the SOA of those domains remotely and locally and creates a list of stale domains. For each of these domains, PDNS starts a zonetransfer to resynchronise. Because zone transfers can fail, it is important that the interface to the backend allows for transaction semantics because a zone might otherwise be left in a halfway updated situation.

The following excerpt from the DNSBackend shows the relevant functions:

```
  class DNSBackend {
  public:
          /* ... */
          virtual bool getDomainInfo(const string &domain, DomainInfo &di);
   virtual bool isMaster(const string &name, const string &ip);
   virtual bool startTransaction(const string &qname, int id);
   virtual bool commitTransaction();
   virtual bool abortTransaction();
   virtual bool feedRecord(const DNSResourceRecord &rr);
   virtual void getUnfreshSlaveInfos(vector<DomainInfo>* domains);
   virtual void setFresh(int id);
          /* ... */
 }
```

The mentioned DomainInfo struct looks like this:

Table C-3. DomainInfo struct

int id	ID of this zone within this backend
string master	IP address of the master of this domain, if any
u_int32_t serial	Serial number of this zone
u_int32_t notified_serial	Last serial number of this zone that slaves have seen
time_t last_check	Last time this zone was checked over at the master for changes
enum {Master,Slave,Native} kind	Type of zone

DNSBackend *backend	Pointer to the backend that feels authoritative for a domain and can act as a slave

These functions all have a default implementation that returns false - which explains that these methods can be omitted in simple backends. Furthermore, unlike with simple backends, a slave capable backend must make sure that the 'DNSBackend *db' field of the SOAData record is filled out correctly - it is used to determine which backend will house this zone.

bool isMaster(const string &name, const string &ip);

If a backend considers itself a slave for the domain **name** and if the IP address in **ip** is indeed a master, it should return true. False otherwise. This is a first line of checks to guard against reloading a domain unnecessarily.

void getUnfreshSlaveInfos(vector<DomainInfo>* domains)

When called, the backend should examine its list of slave domains and add any unfresh ones to the domains vector.

bool getDomainInfo(const string &name, DomainInfo & di)

This is like getUnfreshSlaveInfos, but for a specific domain. If the backend considers itself authoritative for the named zone, `di` should be filled out, and 'true' be returned. Otherwise return false.

bool startTransaction(const string &qname, int id)

When called, the backend should start a transaction that can be committed or rolled back atomically later on. In SQL terms, this function should **BEGIN** a transaction and **DELETE** all records.

bool feedRecord(const DNSResourceRecord &rr)

Insert this record.

bool commitTransaction();

Make the changes effective. In SQL terms, execute **COMMIT**.

bool abortTransaction();

Abort changes. In SQL terms, execute **ABORT**.

bool setFresh()

Indicate that a domain has either been updated or refreshed without the need for a retransfer. This causes the domain to vanish from the vector modified by `getUnfreshSlaveInfos()`.

PDNS will always call `startTransaction()` before making calls to `feedRecord()`. Although it is likely that `abortTransaction()` will be called in case of problems, backends should also be prepared to abort from their destructor.

The actual code in PDNS is currently (1.99.9):

```
    Resolver resolver;
    resolver.axfr(remote,domain.c_str());

    db->startTransaction(domain, domain_id);

    L<<Logger::Error<<"AXFR started for '"<<domain<<"'"<<endl;
    Resolver::res_t recs;

    while(resolver.axfrChunk(recs)) {
      for(Resolver::res_t::const_iterator i=recs.begin();i!=recs.end();++i) {
db->feedRecord(*i);
      }
    }
    db->commitTransaction();
    db->setFresh(domain_id);
    L<<Logger::Error<<"AXFR done for '"<<domain<<"'"<<endl;
```

C.4.1. Supermaster/Superslave capability

A backend that wants to act as a 'superslave' for a master should implement the following method:

```
class DNSBackend
{
   virtual bool superMasterBackend(const string &ip, const string &domain, cons
};
```

This function gets called with the IP address of the potential supermaster, the domain it is sending a notification for and the set of NS records for this domain at that IP address.

Using the supplied data, the backend needs to determine if this is a bonafide 'supernotification' which should be honoured. If it decides that it should, the supplied pointer to 'account' needs to be filled with the configured name of the supermaster (if accounting is desired), and the db needs to be filled with a pointer to your backend.

Supermaster/superslave is a complicated concept, if this is all unclear see Section 13.2.1.

C.5. Read/write master-capable backends

In order to be a useful master for a domain, notifies must be sent out whenever a domain is changed. Periodically, PDNS queries backends for domains that may have changed, and sends out notifications for slave nameservers.

In order to do so, PDNS calls the `getUpdatedMasters()` method. Like the `getUnfreshSlaveInfos()` function mentioned above, this should add changed domain names to the vector passed.

The following excerpt from the DNSBackend shows the relevant functions:

```
class DNSBackend {
public:
         /* ... */
 virtual void getUpdatedMasters(vector<DomainInfo>* domains);
 virtual void setNotifed(int id, u_int32_t serial);
         /* ... */
}
```

These functions all have a default implementation that returns false - which explains that these methods can be omitted in simple backends. Furthermore, unlike with simple backends, a slave capable backend must make sure that the 'DNSBackend *db' field of the SOAData record is filled out correctly - it is used to determine which backend will house this zone.

void getUpdatedMasters(vector<DomainInfo>* domains)

When called, the backend should examine its list of master domains and add any changed ones to the DomainInfo vector

bool setNotified(int domain_id, u_int32_t serial)

Indicate that notifications have been queued for this domain and that it need not be considered 'updated' anymore

Appendix D. Compiling PowerDNS

D.1. Compiling PowerDNS on Unix

> **Note:** For now, see the Open Source PowerDNS site (http://www.powerdns.org). **./configure ; make ; make install** will do The Right Thing for most people.

PowerDNS can becompiled with modules built in, or with modules designed to be loaded at runtime. All that is configured before compiling using the well known autoconf/automake system.

To compile in modules, specify them as **--with-modules="mod1 mod2 mod3"**, substituting the desired module names. Each backend has a module name in the table at the beginning of its section.

To compile a module for inclusion at runtime, which is great if you are a unix vendor, use **--with-dynmodules="mod1 mod2 mod3"**. These modules then end up as .so files in the compiled libdir.

D.1.1. AIX

Known to compile with gcc, but only since 2.9.8. AIX lacks POSIX semaphores so they need to be emulated, as with MacOS X.

D.1.2. FreeBSD

Works fine, but use gmake. Pipe backend is currently broken, for reasons, see Section A.1. Due to the threading model of FreeBSD, PowerDNS does not benefit from additional CPUs on the system.

D.1.3. Linux

Linux is probably the best supported platform as most of the main coders are Linux users. The static DEB distribution is known to have problems on Debian 'Sid', but that doesn't matter as PowerDNS is a native part of Debian 'Sid'. Just apt-get!

D.1.4. MacOS X

Did compile at one point but maintenance has lapsed. Let us know if you can provide us with a login on MacOS X or if you want to help.

D.1.5. OpenBSD

Compiles but then does not work. We hear that it may work with more recent versions of gcc, please let us know on <pdns-dev@mailman.powerdns.com>.

D.1.6. Solaris

Solaris 7 is supported, but only just. AAAA records do not work on Solaris 7. Solaris 8 and 9 work fine. The 'Sunpro' compiler has not been tried but is reported to be lacking large parts of the Standard Template Library, which PowerDNS relies on heavily. Use gcc and gmake (if available). Regular Solaris make has some issues with some PowerDNS Makefile constructs.

D.2. Compiling PowerDNS on Windows

By Michel Stol (<michel@powerdns.com>).

D.2.1. Assumptions

I will assume these things from you:

You have the PowerDNS sources.

There's not much to compile without the source files, eh? :)

You are using Microsoft Visual C++. If you get it to compile using a free compiler, please let us know!

From the day that we began porting the UNIX PowerDNS sources to Microsoft Windows we used Microsoft Visual C++ as our development environment of choice.

We used Visual C++ 6.0 to compile all sources (both standard version and SP5). Other versions (including Visual C++ .NET) are untested.

You are using Microsoft Windows NT, 2000 or XP

I will assume that the system where you want to compile the sources on is running Microsoft Windows NT, 2000 or XP. These are the operating systems that where found running PowerDNS for Windows.

Note: You probably can compile the sources on other Windows versions too, but that is currently untested.

You are using an English Windows version.

Troughout this document I will use the English names for menu items, names etc., so if you are running a non-English Windows or MSVC version you have to translate those things yourself. But I don't think that would be a big problem.

D.2.2. Prequisites

Although we tried to keep PowerDNS for Windows' dependencies down to a minimum, you will still need some programs and libraries to be able to compile the sources.

D.2.2.1. pthreads for Windows

The pthreads for Windows library is a Windows implementation of the POSIX threads specification, which is used a lot in UNIX programs.

PowerDNS uses pthreads too, and to ease the porting process we decided not to reinvent the wheel, but to use pthreads for Windows instead.

D.2.2.1.1. Getting pthreads for Windows

Pthreads for Windows is available from anonymous ftp at ftp://sources.redhat.com/pub/pthreads-win32/. You should download the latest `pthreads-YYYY-MM-DD.exe` file.

Note: PowerDNS for Windows was tested with the snapshot of 2002-03-02 of the library.

For more information you can visit the pthreads for Windows homepage at http://sources.redhat.com/pthreads-win32/

D.2.2.2. Installing pthreads for Windows

To install the pthreads for Windows library you have to locate your `pthreads-YYYY-MM-DD.exe` file and start it.

After starting the executable a self-extractor dialog will show up where you can specify where to extract the contents of the file. When you selected a location you can press the Extract button to extract all content to the target directory.

The library is now installed, we still have to tell Visual C++ where it's located though, more on that later.

D.2.3. Nullsoft Installer

For our installation program we used Nullsoft's Installer System (NSIS). We used NSIS because it's easy to use, versatile and free (and it uses SuperPiMP™ technology, but they refuse to tell us what it is ;)). If the name Nullsoft rings a bell, it's because they're the guys who made winamp (http://www.winamp.com/).

D.2.3.1. Getting the Nullsoft Installer

The Nullsoft Installer can be downloaded at their website, which is located at http://www.nullsoft.com/free/nsis/. The file that you should download is called `nsisXXX.exe` (where XXX is the latest version).

> **Note:** You can find the NSIS documentation at that website too.

D.2.3.2. Installing the Nullsoft Installer

Installing NSIS is easy. All there is to it is locating the installer and execute it. Then just follow the installation steps.

D.2.4. Setting up the build-environment

Before starting Microsoft Visual C++ and compile PowerDNS for Windows, you first have to set up your build environment.

D.2.4.1. Make Microsoft Visual C++ recognize `*.cc` and `*.hh` (optional)

All PowerDNS source files are in the form `name.cc`, and all header files in the form `name.hh`. These extensions aren't recognized by MSVC by default, so you might want to change that first.

> **Note:** Only perform this step if you want to be able to edit the `*.cc` and `*.hh` files in MSVC.

> **Caution**
>
> If you decide to perform this step, remember that it requires modification of the Windows registry, always make a backup before modifying!

Ok, after that word of caution we can now proceed. You have to follow these steps:

1. Start the registry editor by entering `regedit.exe` in the run prompt (Start->Run...).
2. Right click on `HKEY_CLASSES_ROOT` and select New->Key. A new key will appear, change that key to "`.cc`", then change the default value to "cppfile"

   Then perform the same step for "`.hh`" (use "hfile" instead of "cppfile").

3. Go to `HKEY_CURRENT_USER\Software\Microsoft\DevStudio\6.0\Build System\Components\Platforms\Win32 (x86)\Tools\32-bit C/C++ Compiler for 80x86`. And add ";*.cc" to the `Input_Spec` value (so that it becomes "*.c;*.cpp;*.cxx;*.cc").

   > **Note:** If you happen to use another platform (like alpha) to compile the sources, you have to do the step above for that platform.

4. Go to `HKEY_CURRENT_USER\Software\Microsoft\DevStudio\6.0\Search`. And add ";*.cc;*.hh" to the `FIF_Filter` value (so that it becomes "*.c;*.cpp;*.cxx;*.tli;*.h;*.tlh;*.inl;*.rc;*.cc;*.hh").
5. Finally change `HKEY_CURRENT_USER\Software\Microsoft\DevStudio\6.0\Text Editor\Tabs/Language Settings\C/C++`. And add ";cc;hh" to the `FileExtensions` value (so that it becomes "cpp;cxx;c;h;hxx;hpp;inl;tlh;tli;rc;rc2;hh;cc").
6. Close the registry editor.

Now should MSVC properly recognize the files as being C++.

D.2.4.2. Setting Microsoft Visual C++'s directories

MSVC needs to locate some include files, libraries and executables when it has to build PowerDNS for Windows. We are now going to tell MSVC where to find those.

To enter the directory dialog you have to go to Tools->Options...->Directories.

D.2.4.2.1. Setting the pthreads directories

When you are in the directory dialog you can add the pthreads for Windows directory.

First add the include directory, to do this you have to select Include files from the Show directories for: combobox. Then press the New button and browse to the *include* directory of pthreads (ie. `C:\pthreads\include`).

Then switch to Library files and add the *library* directory (ie. `C:\pthreads\lib`) using the same method as above.

D.2.4.2.2. Setting the Nullsoft Installer directory

While still being in the directory dialog, switch to Executable files and add the Nullsoft Installer directory (ie. `C:\Program Files\NSIS`) to the list.

D.2.5. Compilation

Finally, after all the reading, installing and configuring we are ready to start compiling PowerDNS for Windows.

D.2.5.1. Starting the compilation

To start the compilation you first have to open the PowerDNS workspace (`powerdns.dsw`) using explorer or from the File->Open Workspace... menu in MSVC.

After you opened the workspace you can start compiling. Check all the checkboxes in the Build->Batch Build... menu and press the Build button.

Now cross your fingers and go make some coffee or tea while compiling PowerDNS for Windows. :)

D.2.5.2. Yay! It compiled

Congratulations, you have now compiled PowerDNS for Windows!

All the release builds of the binaries are in the `Release` directory (including the generated installer). The debug builds are in the, guess what, `Debug` directory.

Now you can start installing PowerDNS, but that's beyond the scope of this document. See the online documentation (http://downloads.powerdns.com/documentation/html/) for more information about that.

D.2.5.3. What if it went wrong?

If the compilation fails, then try reading this article again, and again to see if you did something wrong.

If you are pretty sure that it's a bug, either in the PowerDNS sources, the build system or in this article, then please send an e-mail to <`pdns-dev@mailman.powerdns.com`> describing your problem. We will then try to fix it.

D.2.6. Miscellaneous

Some miscellaneous information.

D.2.6.1. Credits

Michel Stol would like to thank these people:

Bert Hubert

For writing the wonderfull PowerDNS software and learning me stuff that I'd otherwise never had learned.

PowerDNS B.V.

For being great colleagues.

The pthreads-win32 crew (see the pthreads-win32 `CONTRIBUTORS` file).

For easing our porting process by writing a great Windows implementation of pthreads.

The guys over at Nullsoft.

For creating the Nullsoft Installer System (NSIS), and Winamp, the program we use every day to make a lot of noise in the office.

D.2.6.2. Contact information

If you have a comment, or a bug report concerning either this document or the PowerDNS sources you can contact <pdns-dev@mailman.powerdns.com>

For general information about PowerDNS, the pdns server, express, documentation etc. I advice you to visit http://www.powerdns.com/

If you are interested in buying PowerDNS you can send a mail to <sales@powerdns.com> or you can visit the PowerDNS website at http://www.powerdns.com/pdns/

If you want to praise my work, ask me to marry you, deposit $1.000.000 on my bank account or flame me to death, then you can mail me at <michel@powerdns.com> :)

D.2.6.3. Legal information

Appendix E. PowerDNS license (GNU General Public License version 2)

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you

provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code. 4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License. 8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is

permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS