

PowerDNS manual

Contents

1	The PowerDNS dynamic nameserver	1
1.1	Function & design of PDNS	1
1.2	About this document	1
1.3	Release notes	2
1.3.1	PowerDNS Authoritative Server 3.1	2
1.3.2	Authoritative Server version 2.9.22.6	7
1.3.3	Authoritative Server version 2.9.22.5	7
1.3.4	PowerDNS Authoritative Server 3.0.1	8
1.3.5	PowerDNS Authoritative Server 3.0	8
1.3.6	Recursor version 3.3.1	13
1.3.7	Recursor version 3.3	14
1.3.8	Recursor version 3.2	15
1.3.9	Recursor version 3.1.7.2	19
1.3.10	Recursor version 3.1.7.1	19
1.3.11	Authoritative Server version 2.9.22	20
1.3.12	Authoritative Server version 2.9.21.2	23
1.3.13	Authoritative Server version 2.9.21.1	23
1.3.14	Recursor version 3.1.7	23
1.3.15	Recursor version 3.1.6	24
1.3.16	Recursor version 3.1.5	24
1.3.17	PowerDNS Authoritative Server version 2.9.21	27
1.3.18	Recursor version 3.1.4	29
1.3.19	Recursor version 3.1.3	30
1.3.20	Recursor version 3.1.2	31
1.3.21	Recursor version 3.1.1	32
1.3.22	Recursor version 3.0.1	33
1.3.23	Recursor version 3.0	34
1.3.24	Version 2.9.20	35
1.3.25	Version 2.9.19	36
1.3.26	Version 2.9.18	38

1.3.27	Version 2.9.17	40
1.3.28	Version 2.9.16	41
1.3.29	Version 2.9.15	42
1.3.30	Version 2.9.14	42
1.3.31	Version 2.9.13	43
1.3.32	Version 2.9.12	44
1.3.33	Version 2.9.11	45
1.3.34	Version 2.9.10	45
1.3.35	Version 2.9.8	46
1.3.36	Version 2.9.7	46
1.3.37	Version 2.9.6	47
1.3.38	Version 2.9.5	48
1.3.39	Version 2.9.4	49
1.3.40	Version 2.9.3a	50
1.3.41	Version 2.9.2	52
1.3.42	Version 2.9.1	53
1.3.43	Version 2.9	54
1.3.44	Version 2.8	54
1.3.45	Version 2.7 and 2.7.1	55
1.3.46	Version 2.6.1	55
1.3.47	Version 2.6	55
1.3.48	Version 2.5.1	56
1.3.49	Version 2.5	56
1.3.50	Version 2.4	58
1.3.51	Version 2.3	58
1.3.52	Version 2.2	59
1.3.53	Version 2.1	59
1.3.54	Version 2.0.1	60
1.3.55	Version 2.0	60
1.3.56	Version 2.0 Release Candidate 2	61
1.3.57	Version 2.0 Release Candidate 1	61
1.3.58	Version 1.99.12 Prerelease	62
1.3.59	Version 1.99.11 Prerelease	63
1.3.60	Version 1.99.10 Prerelease	63
1.3.61	Version 1.99.9 Early Access Prerelease	64
1.3.62	Version 1.99.8 Early Access Prerelease	65
1.3.62.1	Known bugs	65
1.3.62.2	Missing features	65
1.3.63	Version 1.99.7 Early Access Prerelease	66

1.3.63.1	Known bugs	66
1.3.63.2	Missing features	66
1.3.64	Version 1.99.6 Early Access Prerelease	66
1.3.64.1	Known bugs	67
1.3.64.2	Missing features	67
1.3.65	Version 1.99.5 Early Access Prerelease	67
1.3.65.1	Known bugs	67
1.3.65.2	Missing features	67
1.3.66	Version 1.99.4 Early Access Prerelease	68
1.3.66.1	Known bugs	68
1.3.66.2	Missing features	68
1.3.67	Version 1.99.3 Early Access Prerelease	68
1.3.67.1	Known bugs	69
1.3.67.2	Missing features	69
1.3.68	Version 1.99.2 Early Access Prerelease	69
1.3.68.1	Known bugs	70
1.3.68.2	Missing features	70
1.3.69	Version 1.99.1 Early Access Prerelease	70
1.3.69.1	Known bugs	70
1.3.69.2	Missing features	71
1.4	Security	71
1.5	PowerDNS Security Advisory 2006-01: Malformed TCP queries can lead to a buffer overflow which might be exploitable	72
1.6	PowerDNS Security Advisory 2006-02: Zero second CNAME TTLs can make PowerDNS exhaust allocated stack space, and crash	72
1.7	PowerDNS Security Advisory 2008-01: System random generator can be predicted, leading to the potential to 'spooof' PowerDNS Recursor	72
1.8	PowerDNS Security Advisory 2008-02: By not responding to certain queries, domains become easier to spoof	74
1.9	PowerDNS Security Advisory 2008-02: Some PowerDNS Configurations can be forced to restart remotely	74
1.10	PowerDNS Security Advisory 2010-01: PowerDNS Recursor up to and including 3.1.7.1 can be brought down and probably exploited	75
1.11	PowerDNS Security Advisory 2010-02: PowerDNS Recursor up to and including 3.1.7.1 can be spoofed into accepting bogus data	75
1.12	PowerDNS Security Advisory 2012-01: PowerDNS Authoritative Server can be caused to generate a traffic loop	76
1.13	Acknowledgements	77
2	Installing on Unix	78
2.1	Possible problems at this point	78
2.2	Testing your install	78
2.2.1	Typical errors	79
2.3	Running PDNS on unix	79

3	Installing on Microsoft Windows	81
3.1	Configuring PDNS on Microsoft Windows	82
3.2	Running PDNS on Microsoft Windows	82
4	Basic setup: configuring database connectivity	83
4.1	Example: configuring MySQL	84
4.1.1	Common problems	86
5	Dynamic resolution using the PipeBackend	87
5.1	Deploying the PipeBackend with the BindBackend	87
6	Logging & Monitoring Authoritative Server performance	88
6.1	Webserver	88
6.2	Via init.d commands	88
6.3	Operational logging using syslog	90
7	Security settings & considerations	91
7.1	Settings	91
7.1.1	Running as a less privileged identity	91
7.1.2	Jailing the process in a chroot	91
7.2	Considerations	92
8	Virtual hosting	93
9	Authoritative Server Performance	94
9.1	General advice	94
9.2	Native Posix Thread Library vs LinuxThreads	94
9.3	Performance related settings	94
9.3.1	Packet Cache	95
9.3.2	Query Cache	95
10	Migrating to PowerDNS	96
10.1	Zone2sql	96
11	Notes on upgrading	98
11.1	From PowerDNS Authoritative Server 2.9.x to 3.0	98
11.1.1	Frequently Asked Questions about 3.0	99
11.2	From PowerDNS Authoritative Server 3.0 to 3.1	99

12	Serving authoritative DNSSEC data	100
12.1	A brief introduction to DNSSEC	100
12.2	Profile, Supported Algorithms, Record Types & Modes of operation	101
12.2.1	DNSSEC: live-signed vs orthodox 'pre-signed' mode	102
12.3	Migration	102
12.3.1	From an existing PowerDNS installation	102
12.3.2	From existing non-DNSSEC non-PowerDNS setups	102
12.3.3	From existing DNSSEC non-PowerDNS setups, pre-signed	102
12.3.4	From existing DNSSEC non-PowerDNS setups, live signing	103
12.4	Records, Keys, signatures, hashes within PowerDNSSEC in online signing mode	103
12.4.1	(Hashed) Denial of Existence	103
12.4.2	Signatures	103
12.5	'pdnssec' for PowerDNSSEC command & control	104
12.6	DNSSEC advice & precautions	105
12.6.1	Packet sizes, fragments, TCP/IP service	105
12.7	Operational instructions	106
12.7.1	Publishing a DS	106
12.7.2	ZSK rollover	106
12.7.3	KSK rollover	106
12.7.4	Going insecure	106
12.7.5	NSEC(3) change	106
12.8	Modes of operation	106
12.8.1	PowerDNSSEC Pre-signed records	106
12.8.2	PowerDNSSEC Front-signing	106
12.8.3	PowerDNSSEC BIND-mode operation	107
12.8.4	PowerDNSSEC hybrid BIND-mode operation	107
12.8.5	Rules for filling out fields in database backends	107
12.9	Security	108
12.10	Performance	108
12.11	Thanks to, acknowledgements	108
13	TSIG: shared secret authorization and authentication	110
13.1	Provisioning outbound AXFR access	110
13.2	Provisioning signed notification and AXFR requests	111
14	AXFR ACLs	112
15	Per zone settings aka Domain Metadata	113
16	Recursion	114
16.1	Details	114

17 PowerDNS Recursor: a high performance resolving nameserver	116
17.1 pdns_recursor settings	116
17.2 pdns_recursor command line	119
17.3 Controlling and querying the recursor	119
17.4 PowerDNS Recursor performance	120
17.4.1 Recursor Caches	121
17.5 Details	121
17.5.1 Anti-spoofing	121
17.5.2 Throttling	122
17.6 Statistics	122
17.7 Scripting	123
17.7.1 Configuring Lua scripts	123
17.7.2 Writing Lua PowerDNS Recursor scripts	124
17.8 Design and Engineering of the PowerDNS Recursor	125
17.8.1 The PowerDNS Recursor	126
17.8.2 Synchronous code using MTasker	126
17.8.3 MPlexer	126
17.8.4 MOADNSParser	127
17.8.5 The C++ Standard Library / Boost	128
17.8.6 Actual DNS Algorithm	128
17.8.7 The non-cached case	129
17.8.8 Some of the things we glossed over	130
17.8.9 The Recursor Cache	130
17.8.10 Some small things	130
18 Master/Slave operation & replication	131
18.1 Native replication	131
18.2 Slave operation	131
18.2.1 Supermaster automatic provisioning of slaves	132
18.2.2 Modifying a slave zone using a script	132
18.3 Master operation	133
19 Fancy records for seamless email and URL integration	135
20 Index of all Authoritative Server settings	136
21 Index of all Authoritative Server metrics	139
21.1 Counters & variables	139
21.1.1 Counters	139
21.1.2 Ring buffers	139

22 Supported record types and their storage	141
23 HOWTO & Frequently Asked Questions	143
23.1 Getting support, free and paid FAQ	143
23.2 Using and Compiling PowerDNS FAQ	143
23.3 Backend developer HOWTO	145
23.4 About PowerDNS.COM BV, 'the company'	146
24 Other tools included with PowerDNS	147
24.1 Notification proxy (nproxy)	147
25 Tools to analyse DNS traffic	148
A Backends in detail	149
A.1 PipeBackend	149
A.1.1 PipeBackend protocol	150
A.1.1.1 Handshake	150
A.1.1.2 Questions	150
A.1.1.3 Answers	151
A.1.1.4 Sample perl backend	152
A.1.2 Notes	152
A.2 Random Backend	153
A.3 Generic MySQL and PgSQL backends	153
A.3.1 MySQL specifics	154
A.3.2 PostgreSQL specifics	155
A.3.3 Oracle specifics	157
A.3.4 Basic functionality	158
A.3.5 DNSSEC queries	159
A.3.6 Master/slave queries	160
A.3.7 Fancy records	160
A.3.8 Settings and specifying queries	161
A.3.9 Native operation	161
A.3.10 Slave operation	161
A.3.11 Superslave operation	162
A.3.12 Master operation	162
A.4 Oracle backend	162
A.4.1 The Database Schema	163
A.4.1.1 Zones Table	163
A.4.1.2 The Zonemasters and ZoneAlsoNotify Tables	164
A.4.1.3 The Supermasters Table	164

A.4.1.4	The ZoneMetadata Table	164
A.4.1.5	The Tables for Cryptographic Keys	164
A.4.1.6	The Records Table	165
A.4.2	The SQL Statements	165
A.4.2.1	Fetching DNS records	165
A.4.2.2	Zone Metadata and TSIG	166
A.4.2.3	DNSSEC	167
A.4.2.4	Incoming AXFR	168
A.4.2.5	Master/Slave Stuff	168
A.4.2.6	Superslave Stuff	169
A.5	Generic SQLite backend (2 and 3)	169
A.5.1	Compiling the SQLite backend	170
A.5.2	Setting up the database	170
A.5.3	Using the SQLite backend	171
A.6	DB2 backend	172
A.7	Bind zone file backend	172
A.7.1	Operation	173
A.7.2	Pdns_control commands	173
A.7.3	Performance	173
A.7.4	Master/slave configuration	173
A.7.4.1	Master	173
A.7.4.2	Slave	173
A.7.5	Commands	174
A.8	ODBC backend	174
A.9	XDB Backend	175
A.10	LDAP backend	175
A.11	OpenDBX backend	176
A.12	Geo backend	176
A.13	MongoDB Backend	177
A.14	Lua Backend	177
A.15	TinyDNS Backend	178
A.15.1	Configuration Parameters	178
A.15.2	Location and Timestamp support	178
A.15.3	Master mode	179
A.15.4	Useful implementation notes	179
B	PDNS internals	180
B.1	Controlsocket	180
B.1.1	pdns_control	180
B.2	Guardian	181
B.3	Modules & Backends	181
B.4	How PDNS translates DNS queries into backend queries	182

C	Backend writers' guide	183
C.1	Simple read-only native backends	183
C.1.1	A sample minimal backend	184
C.1.2	Interface definition	186
C.2	Reporting errors	187
C.3	Declaring and reading configuration details	187
C.4	Read/write slave-capable backends	188
C.4.1	Supermaster/Superslave capability	190
C.5	Read/write master-capable backends	190
D	Compiling PowerDNS	191
D.1	Compiling PowerDNS on Unix	191
D.1.1	AIX	191
D.1.2	FreeBSD	191
D.1.3	Linux	191
D.1.4	MacOS X	192
D.1.5	OpenBSD	192
D.1.6	Solaris	192
D.2	Compiling PowerDNS on Windows	192
D.2.1	Assumptions	192
D.2.2	Prerequisites	192
D.2.2.1	pthreads for Windows	193
D.2.2.1.1	Getting pthreads for Windows	193
D.2.2.1.2	Installing pthreads for Windows	193
D.2.3	Nullsoft Installer	193
D.2.3.1	Getting the Nullsoft Installer	193
D.2.3.2	Installing the Nullsoft Installer	193
D.2.4	Setting up the build-environment	193
D.2.4.1	Make Microsoft Visual C++ recognize *.cc and *.hh (optional)	194
D.2.4.2	Setting Microsoft Visual C++'s directories	194
D.2.4.2.1	Setting the pthreads directories	194
D.2.4.2.2	Setting the Nullsoft Installer directory	195
D.2.5	Compilation	195
D.2.5.1	Starting the compilation	195
D.2.5.2	Yay! It compiled	195
D.2.5.3	What if it went wrong?	195
D.2.6	Miscellaneous	195
D.2.6.1	Credits	195
D.2.6.2	Contact information	196
D.2.6.3	Legal information	196

E	PowerDNS license (GNU General Public License version 2)	197
F	Further copyright statements	200
F.1	AES implementation by Brian Gladman	200
G	Cryptographic software and export control	201
G.1	Specific United States Export Control Notes	201

List of Tables

1.1	PowerDNS Security Advisory	72
1.2	PowerDNS Security Advisory	73
1.3	PowerDNS Security Advisory	73
1.4	PowerDNS Security Advisory	74
1.5	PowerDNS Security Advisory	75
1.6	PowerDNS Security Advisory	75
1.7	PowerDNS Security Advisory	76
1.8	PowerDNS Security Advisory	76
22.1	SOA fields	142
A.1	PipeBackend capabilities	149
A.2	Random Backend capabilities	153
A.3	Generic PostgreSQL and MySQL backend capabilities	154
A.4	Oracle backend capabilities	162
A.5	Generic SQLite backend capabilities	170
A.6	DB2 backend capabilities	172
A.7	Bind zone file backend capabilities	173
A.8	ODBC backend capabilities	174
A.9	LDAP backend capabilities	175
A.10	OpenDBX backend capabilities	176
A.11	Geo backend capabilities	176
A.12	MongoDB backend capabilities	177
A.13	Lua backend capabilities	177
A.14	TinyDNS backend capabilities	178
C.1	DNSResourceRecord class	186
C.2	SOAData struct	186
C.3	DomainInfo struct	189

Abstract

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

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 zone files** or be **more dynamic** in nature. Additionally, through use of clever programming techniques, PowerDNS offers very high domain resolution performance.

Prime examples of backends include relational databases, but also (geographical) load balancing and failover algorithms.

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

1.1 Function & design of PDNS

PowerDNS consists of two parts: the Authoritative Server and the Recursor. Other nameservers fully combine these functions, PowerDNS offers them separately, but can mix both authoritative and recursive usage seamlessly. The Authoritative Server 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. Depending on your needs, this backend can either be the PowerDNS recursor or an external one.

When the Authoritative Server 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.

The Recursor, conversely, by default has no knowledge of domains itself, but will always consult other authoritative servers to answer questions given to it.

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/>.

The most up to date version of the HTML documentation can be downloaded as a tar archive from <http://doc.powerdns.com/html.tar.gz>.

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.

Beyond PowerDNS 2.9.20, the Authoritative Server and Recursor are released separately.

1.3.1 PowerDNS Authoritative Server 3.1

**Warning**

Version 3.1 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. There are also some important changes if you are coming from 3.0. Please refer to Section 11.1 and Section 11.2 for important information on correct and stable operation, as well as notes on performance and memory use.

Note

Released on the 4th of May 2012
RC3 released on the 30th of April 2012
RC2 released on the 14th of April 2012
RC1 released on the 23th of March 2012

Note

Downloads:

- [Official download page](#)
 - [CentOS/RHEL 5/6 RPMs](#) kindly provided by Kees Monshouwer.
 - [Additional packages](#) kindly provided by various other people.
-

Version 3.1 of the PowerDNS Authoritative Server represents the 'coming of age' of our DNSSEC implementation. In addition, 3.1 solves a lot of '.0' issues typically associated with a major new release.

As usual, we are very grateful for the involvement of the PowerDNS community. The uptake of 3.0 was rapid, and many users were very helpful in shaking out the bugs, and willing to test the fixes we provided or, in many cases, provided the fixes themselves.

Of specific note is the giant PowerDNS DNSSEC deployment in Sweden by Atomia and Binero. PowerDNS 3.0 now powers over 150000 DNSSEC domains in Sweden, around 95% of all DNSSEC domains, in a country where most internet service providers actually validate all .SE domains.

Finally, this release has benefited a lot from Peter van Dijk joining us, as he has merged a tremendous amount of patches, cleaned up years of accumulated dust in the code, and massively improved our regression testing into a full blown continuous integration setup with full DNSSEC tests!

Additionally, we would like to thank Ruben d'Arco, Jose Arthur Benetasso Villanova, Marc Haber, Jimmy Bergman, Aki Tuomi and everyone else who helped us out!

Changes between RC3 and final:

- pdnssec now honours the default-soa-name setting. Reported by Kees Monshouwer, fixed in [commit 2600](#).

Changes between RC2 and RC3:

- The hidden test-algorithms command for pdnssec now has a little brother 'test-algorithm X'. Code in [commit 2596](#), by Aki Tuomi.
-

- PolarSSL upgraded to 1.1.2 due to weak RSA key generation ([commit 2586](#)). If you created RSA keys with RC1 or RC2 using PolarSSL, please replace them! This upgrade introduced a slowdown; speedup patch in [commit 2593](#).
- It turns out we were using libmysqlclient in a thread-unsafe manner. This issue was reported and painstakingly debugged by Marc Haber. Presumably fixed in [commit 2591](#).
- Updated a bunch of internal counters to be threadsafe. Code in [commit 2579](#).
- NSEC(3) bitmaps can now cover RRtypes above 255. Reported by Michael Braunoeder, patch by Aki Tuomi in [commit 2590](#).
- pdnssec check-zone now reports MBOXFW and URL records (as those are unsupported since 3.0). Reported by Gerwin Krist of Digitalus, patch by Ruben d'Arco. Closes [ticket 446](#).
- The odbcb backend was removed. It only runs on Windows and Windows is unsupported since 3.0. Removal in [commit 2576](#).
- We used to send the chunk length and the actual chunk in two separate writes (often resulting in two separate TCP packets) during outbound AXFR. This confused MSDNS. We now combine those writes. Code in [commit 2575](#).
- The bindbackend can now run without SQLite3, as previously intended. Fix in [commit 2574](#).
- Some high-concurrency master setups would crash under load. Fixed in [commit 2571](#).

Changes between RC1 and RC2:

- We imported the TinyDNS backend by Ruben d'Arco. Code mostly in [commit 2559](#). See Section [A.15](#).
- Overriding C(XX)FLAGS is easier now. Problem pointed out by Jose Arthur Benetasso Villanova and others, fix suggested by Sten Spans. Patch in [commit 2533](#).
- TSIG fixes: skip embedded spaces in keys ([commit 2536](#)), compute signatures correctly (by Ruben d'Arco in [commit 2547](#)),
- nproxy, dnsscan and dnsdemog did not compile at all. Fixes in [commit 2538](#), [commit 2554](#).
- We now allow unescaped tabs in TXT records. Fix in [commit 2539](#).
- SOA records no longer disappear during incoming transfers. Fix by Ruben d'Arco in [commit 2540](#).
- PowerDNS compiles on OS X (and other platforms that support our auth server but not the recursor) again, fix in [commit 2566](#).
- Cleanups related to warnings from gcc and valgrind in [commit 2561](#), [commit 2562](#), [commit 2565](#).
- Solaris compatibility fixes by Ruben d'Arco, Juraj Lutter and others in [commit 2548](#), [commit 2552](#), [commit 2553](#), [commit 2560](#). Fixes for *BSD in [commit 2546](#).
- pdns_control help would report 'version' twice, reported by Gerwin, fix in [commit 2549](#).

DNSSEC related fixes:

- When slaving zones, PowerDNS now automatically detects that a zone is pre-signed. Code in [commit 2502](#), closing [ticket 369](#), [ticket 392](#).
- The bindbackend can now manage its own SQLite3 database to store key data, removing the need to run it with a gsql backend. Code in [commit 2448](#), [commit 2449](#), [commit 2450](#), [commit 2451](#), [commit 2452](#), [commit 2453](#), [commit 2455](#), [commit 2482](#), [commit 2496](#), [commit 2499](#).
- NSEC/NSEC3 logic for picking 'boundary' names was tricky, and got it wrong in some cases. Fixes in [commit 2289](#), [commit 2429](#), [commit 2435](#) and [commit 2473](#).
- The subtle differences between 'what records get NSEC', 'what records get NSEC3' and 'what records should get signed' did not translate well to the SQL auth column. We now use 'ordname IS NULL' to map the whole spectrum. Code in [commit 2477](#), [commit 2480](#), [commit 2492](#).
- Pre-signed AXFR output, although correct, was different from our query responses. Rectified in [commit 2477](#).

- Spotted & fixed by Jimmy Bergman of Atomia, CNAMEs and RRSIGs could have bad interactions. Fix in [commit 2314](#), further refined in [commit 2318](#). Closes [ticket 411](#).
- Spotted & fixed by Jimmy Bergman of Atomia, we now allow direct RRSIG queries even when do=0.
- Spotted by Mark Scholten and Marco Davids, we would sometimes generate duplicate (and wrong) RRSIGs when signing an ANY answer because of record jumbling. Fix in [commit 2381](#).
- Several fixes to handling of DS queries, in [commit 2420](#), [commit 2510](#), [commit 2512](#).
- We now lowercase the signer name in an RRSIG. This is not mandated by DNSSEC specification but it improves compatibility with some validators. Fix in [commit 2426](#).

Bug fixes:

- Winfried Angele discovered we would open an additional backend connection per zone in the BIND backend. This only impacted users with multiple simultaneous backends. Fix in [commit 2253](#), closing [ticket 383](#).
- All versions of max-cache-entries setting had confusing behaviour when set to 0. Now clarified to mean that 0 truly means 0, and not 'infinite'. Change in [commit 2328](#).
- Wildcards in the presence of delegations were broken. Reported by a cast of thousands. Fix & regression test in [commit 2368](#). Closes [ticket 389](#).
- Internal caches used an order of magnitude more memory than expected and some were not purged properly, which hindered real life deployments. Spotted by Winfried Angele and others. Fixed in [commit 2287](#) and [commit 2328](#).
- Christof Meerwald discovered our .tar file missed a file of the Lua backend. Change in [commit 2257](#).
- Paul Xek found out that the edns-subnet support did not work for subnets tinier than a /25 or /121. Fix in [commit 2258](#).
- edns-subnet aware PIPE scripts received bogus remote information on AXFR requests. Fixed in [commit 2284](#).
- Fix compilation against older versions of MySQL that do not have MYSQL_OPT_RECONNECT. [commit 2264](#), closing [ticket 378](#).
- D. Stussy of Snarked.net discovered that PowerDNS could not parse a DNS packet with a trailing blob of unknown length. Fixed in [commit 2267](#).
- 'pdnssec' did not work for records with NULL ttls. Fixed in [commit 2266](#), closing [ticket 432](#).
- Pipe backend had issues parsing IPv6 records in ABI version 3. Fixed in [commit 2260](#).
- We truncated the altitude in LOC records! I hope no one got lost. Fix in [commit 2268](#).
- Xander Soldaat discovered that even if the web server was not configured, we'd still listen on the port. Fix in [commit 2269](#), closes [ticket 402](#).
- The PIPE backend issues frequent fork(s), leading to potential fd leaks if these are not marked as 'close on exec'. Solved in [commit 2273](#), closing [ticket 194](#).
- Robert van der Meulen found that we messed up the interaction between wildcards and CNAMEs. Fixed in [commit 2276](#), which also adds a regression test to prevent this issue from recurring.
- Fred Wittekind discovered that our notification proxy 'nproxy' no longer built from source. Fixed in [commit 2278](#).
- Grant Keller found that we were inconsistent with spaces in labels, thus breaking DNS-SD. Fix in [commit 2305](#).
- Winfried Angele fixed our autoconf script for Lua detection in [commit 2308](#).
- BIND backend would leak an fd when including a configuration file from named.conf. Spotted by Hannu Ylitalo of Nebula Oy in [commit 2359](#).
- SQLite3 backend could crash on a network error at the wrong moment, leading to a restart by the guardian. Fix in [commit 2336](#).

- `./configure --enable-verbose-logging` was broken, fixed in [commit 2312](#).
- PowerDNS would serve up old SOA data immediately after sending out a notification. Complicated bug documented perfectly in [ticket 427](#), which also came with not one but with two different patches to fix the problem. Thanks to Keith Buck. Code in [commit 2408](#).
- Flag `--start-id` in `zone2sql` was not functional. Removed for now in [commit 2387](#), closing [ticket 332](#).
- Our distribution tarball did not have the SQL schemas. Fixed in [commit 2459](#) and [commit 2460](#).
- "Empty" MX records would confuse one of our parsers. Fixed in [commit 2468](#), closing Debian bug 533023.
- The `pdns.conf` `wildcards`-setting did not do anything in 3.0, so it was removed. Change in [commit 2508](#), [commit 2509](#).
- Additional processing based on records loaded by the BIND backend might fail because of a trailing dot mismatch. Fix in [commit 2398](#).

New features:

- Per-zone AXFR ACLs, based on the `allow-axfr-ips` zone metadata item. Code in [commit 2274](#). Also, remove some remains of our previous approach to supporting this in [commit 2326](#).
- Alberto Donato and Zsolt Dollenstein implemented autoserial support for the Generic SQL backends. Code in [commit 2290](#), [commit 2294](#), [commit 2296](#), [commit 2299](#), [commit 2300](#), [commit 2303](#). Closes [ticket 52](#), [ticket 299](#), [ticket 301](#), [ticket 336](#).
- New SOA Serial Tweak mode INCEPTION-EPOCH for when operating as a 'signing slave', contributed by Jimmy Bergman. Code and documentation in [commit 2320](#).
- Newlines in the `'content'` field of backends are now allowed, restoring some DKIM setups to working condition. Update in [commit 2394](#), closing [ticket 395](#).

Improvements:

- Depending on the encoding used, MySQL could take issue with our `'tsigkeys'` table which contained very large rows. Trimmed in [commit 2400](#), closing [ticket 410](#).
- Various build/configure-related fixes in [commit 2319](#), [commit 2373](#), [commit 2386](#), closing [ticket 380](#), [ticket 405](#), [ticket 420](#).
- We now show the SOA serial after zone transfers. Code in [commit 2385](#), closing [ticket 416](#).
- Ruben d'Arco submitted a full rework of our slave-side AXFR TSIG handling, closing [ticket 393](#) and [ticket 400](#) in the process. Code in [commit 2506](#). Additional improvement in [commit 2513](#).
- The `records.name`-column in the `gpgsql` schema is now constrained to lowercase, as PowerDNS would be unable to find other entries anyway. Fix in [commit 2503](#), closing [ticket 426](#).
- The `gsql`-backends can now handle huge records, thanks to a patch by Ruben d'Arco. Code in [commit 2476](#), closing [ticket 407](#). Additional changes in [commit 2292](#), [commit 2487](#), [commit 2489](#). Closes [ticket 218](#), [ticket 316](#).
- Some of PowerDNS' internal classes would work with uninitialized data when repurposed outside of the PowerDNS core logic. Fix in [commit 2469](#),
- `pdnssec` now has `'check-all-zones'` and `'rectify-all-zones'` commands. Submitted by Ruben d'Arco, code in [commit 2467](#).
- `'restart'` in our `init.d`-script would not start `pdns` if it was down before. Fixed in [commit 2462](#).
- `'pdnssec rectify-zone'` now honours `--verbose` and is rather quiet without it. Code in [commit 2443](#).
- Improved error messages for systems without IPv6. Changes in [commit 2425](#).
- The `packet-` and `querycache` now honour TTLs from backend data. Code in [commit 2414](#).
- `'pdns_control help'` now shows useful usage information. Code in [commit 2410](#) and [commit 2465](#).

-
- Jasper Spaans improved our init.d script for compliance with Debian Squeeze. Patch in [commit 2251](#). Further improvement with 'set -e' to initscript contributed by Marc Haber in [commit 2301](#).
 - Klaus Darilion discovered our configuration file template and --help output explained the various cache TTLs wrongly, and he also added documentation for some missing parameters. [commit 2271](#) and [commit 2272](#).
 - Add support for building against Botan 1.10 (stable) and drop support for 1.9 (development). Changes in [commit 2334](#). This fixes several bugs when building against 1.9.
 - Upgrade internal PolarSSL library to their version 1.1.1. Change in [commit 2389](#) and beyond.
 - Compilation of several backends failed for Boost in non-standard locations. Fixes in [commit 2316](#).
 - We now do additional processing for SRV records too. Code in [commit 2388](#), closing [ticket 423](#) (which also contained the patch). Regression test updates that flow from this in [commit 2390](#).
 - Fix compilation on OSX. [commit 2316](#).
 - Fix pdnssec crash when asked to do DNSSEC without a DNSSEC capable backend. Code in [commit 2369](#).
 - If PowerDNS was not configured to operate as a DNS master, it would still accept 'pdns_control notify' commands, but then not do it. Spotted by David Gavarret, patch by Jose Arthur Benetasso Villanova in [commit 2379](#).
 - In various places we would only accept UPPERCASE DNS typenames. Fixed in [commit 2370](#), closing [ticket 390](#).
 - We would not always drop supplemental groups correctly. Reported by David Black of Atlassian.
 - Our regression tests have been strengthened a lot, and now cover way more features. Commits in [2280](#), [2281](#), [2282](#), [2317](#), [2348](#), [2349](#), [2350](#), [2351](#) and beyond.
 - Update to support the latest draft of DANE/TLSA. Spotted by James Cloos ([commit 2338](#)). Further improvements by Pieter Lexis in [commit 2347](#), [commit 2358](#).
 - Compilation on OpenBSD was eased by patches from Brad Smith, which can be found in [commit 2288](#) and [commit 2291](#), closing [ticket 95](#).
 - 'make check' failed on the internal PolarSSL. Spotted by Daniel Briley, fix in [commit 2283](#).
 - The default SQL schemas were expanded to contain far longer content fields. [commit 2292](#), [commit 2293](#).
 - Documentation typos, Jake Spencer ([commit 2304](#)), Jose Arthur Benetasso Villanova ([commit 2337](#)). Code typos in [commit 2324](#) (closes [ticket 296](#)).
 - Manpage updates from Debian, provided by Matthijs Möhlmann. Content in [commit 2306](#).
 - pdnssec rectify-zone can now accept multiple zones at the same time. Code in [commit 2383](#).
 - As suggested in [ticket 416](#), we now log the SOA serial number after committing an AXFRred zone to the backend. Code in [commit 2385](#).
 - Pick up location of sqlite3 libraries using pkg-config. Implemented using a variation of the patch found in the, now closed, [ticket 380](#). Code in [commit 2386](#).
 - Documented 'pdnssec --verbose' flag is now accepted. Code in [commit 2384](#), closing [ticket 404](#).
 - 'pdnssec --help' now lists all supported signing algorithms. Suggested by Jose Arthur Benetasso Villanova.
 - PIPE backend example script with edns-subnet support was improved to actually use edns-subnet field. Plus update PIPE backend documentation. Code in [commit 2285](#), more documentation regarding MX and SRV in [commit 2313](#).
 - edns-subnet fields now also output in logfile when available ([commit 2321](#)).
 - When running with virtualized configuration files, we now allow dashes in the configuration name. Suggested by Marc Haber, code in [commit 2295](#). Further fixes by Brielle Bruns in [commit 2327](#).
 - Compilation fixes for GNU/Hurd in [commit 2307](#) via Matthijs Möhlmann.
-

- Marc Haber improved our Debian packaging scripts for smoother upgrades. Code in [commit 2315](#).
- When failing to bind to an IP address, report to which one it failed. [commit 2325](#).
- Supermaster checks were performed synchronously, leading to the possibilities of slowdowns. Fixed in [commit 2402](#).

Other changes:

- Removed the deprecated non-generic mysqlbackend, in [commit 2488](#), [commit 2514](#), [commit 2515](#).
- Removed the deprecated 'pdnsbackend', in [commit 2490](#), [commit 2516](#).
- Removed GRANT statements from the gpgsql schema, as we can't assume they will work for everyone. Change in [commit 2493](#).

Tickets closed but not associated with a commit:

- [ticket 125](#): "PowerDNS offers wild card info. when it is not queried for."
- [ticket 219](#): "Accept NOTIFY from masters on non-standard port"
- [ticket 247](#): "pdns caching weirdness with recursion-desired flag"
- [ticket 253](#): "bind backend crashes on long comment line in included file"
- [ticket 271](#): "PowerDNS Server responding with out-of-zone authority section in case there is a cname"
- [ticket 304](#): "also-notify option for pdns, also gives also-notify for bindbackend."
- [ticket 311](#): "PowerDNSSEC responding with SERVFAIL upon IN A query for a CNAME"
- [ticket 325](#): "CNAME working strange!"
- [ticket 376](#): "Unable to create long TXT records"
- [ticket 412](#): "--without-lua doesn't disable lua"
- [ticket 415](#): "Signing thread died during AXFR of signed domain"
- [ticket 422](#): "ecdsa256 keys bug"

1.3.2 Authoritative Server version 2.9.22.6

The improvements to the master/slave engine in 2.9.22.5 contained one serious bug that can cause crashes on busy setups. 2.9.22.6 fixes this crash.

1.3.3 Authoritative Server version 2.9.22.5

2.9.22.5 is an interim release for those not yet ready to make the jump to 3.0, but do need a more recent version of the Authoritative Server. It also contains the patch from Section [1.12](#)

- Improved performance of master/slave engine, especially when hosting tens or hundreds of thousands of slave zones. Code in commits [1657](#), [1658](#), [1661](#) (which also brings multi-master support), [1662](#) (non-standard ports for masters), [1664](#), [1665](#), [1666](#), [1667](#), [1672](#), [1673](#), [2063](#)).
 - Compilation fixes for more modern compilers ([commit 1660](#), [commit 1694](#))
 - Don't crash on communication error with pdns_control ([commit 2015](#)).
 - Packet cache fixes for UltraSPARC ([commit 1663](#))
 - Fix crashes in the BIND backend ([commit 1693](#), [commit 1692](#))
-

1.3.4 PowerDNS Authoritative Server 3.0.1

3.0.1 consists of 3.0, plus the patch from Section [1.12](#)

1.3.5 PowerDNS Authoritative Server 3.0

**Warning**

Version 3.0 of the PowerDNS Authoritative Server is a major upgrade. Please refer to Section [11.1](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Known issues as of RC3 include:

- Not all new features are fully documented yet
-

Note

Released on the 22nd of July 2011

RC1 released on the 4th of April 2011

RC2 released on the 19th of April 2011

RC3 released on the 19th of July 2011

Version 3.0 of the PowerDNS Authoritative Server brings a number of important features, as well as over two years of accumulated bug fixing.

The largest news in 3.0 is of course the advent of DNSSEC. Not only does PowerDNS now (finally) support DNSSEC, we think that our support of this important protocol is among the easiest to use available. In addition, all important algorithms are supported.

Complete detail can be found in Chapter [12](#). The goal of 'PowerDNSSEC' is to allow existing PowerDNS installations to start serving DNSSEC with as little hassle as possible, while maintaining performance and achieving high levels of security.

Tutorials and examples of how to use DNSSEC in PowerDNS can be found linked from <http://powerdnssec.org>.

PowerDNS Authoritative Server 3.0 development has been made possible by the financial and moral support of:

- [AFNIC, the French registry](#)
- [IPCom's RcodeZero Anycast DNS](#), a subsidiary of NIC.AT, the Austrian registry
- [SIDN, the Dutch registry](#)
- .. (awaiting details) ..

This release has received exceptional levels of community support, and we'd like to thank the following people in addition to those mentioned explicitly below: Peter Koch (DENIC), Olaf Kolkman (NLNetLabs), Wouter Wijngaards (NLNetLabs), Marco Davids (SIDN), Markus Travaile (SIDN), Leen Besselink, Antoin Verschuren (SIDN), Olafur Guðmundsson (IETF), Dan Kaminsky (Recursion Ventures), Roy Arends (Nominet), Miek Gieben (SIDN), Stephane Bortzmeyer (AFNIC), Michael Braunoeder (nic.at), Peter van Dijk, Maik Zumstrull, Jose Arthur Benetasso Villanova (Locaweb), Stefan Schmidt, Roland van Rijswijk (Surfnet), Paul Bakker (Brainspark/Fox-IT), Mathew Hennessy, Johannes Kuehrer (Austrian World4You GmbH), Marc van de Geijn (bHosted.nl), Stefan Arentz and Martin van Hensbergen (Fox-IT), Christof Meerwald, Detlef Peeters, Jack Lloyd, Frank Altpeter, Fredrik Danerklint, Vasilij G Tolstov, Brielle Bruns, Evan Hunt, Ralf van der Enden, Marc Laros, Serge Belyshev, Christian Hofstaedtler, Charlie Smurthwaite, Nikolaos Milas, ..

Changes between RC3 and final:

- Slight tweak to the pipebackend to ease DNSSEC operations ([commit 2239](#), [commit 2247](#)). Also fix pipebackend support in pdnssec tool ([commit 2244](#)).
-

- Upgrade the experimental native Lua backend to the latest version from Fredrik Danerklint ([commit 2240](#)) and include this backend in the .deb packages ([commit 2242](#))
- Remove IPv6 dependency, it was only possible to run master/slave operations on a server with at least one IPv6 address. Some very old virtualized setups turned out to have no IPv6 at all. Fix in [commit 2246](#).

Changes between RC2 and RC3:

- PowerDNS Authoritative Server could not be configured to use an IPv6 based resolving backend. Solved in [commit 2191](#).
- LDAP backend reconfigured the timezone (TZ) setting of the daemon, leading to confusing logfile entries. Fixed by Christian Hofstaedtler in [commit 2913](#), closing [ticket 313](#).
- Non-DNSSEC capable backends could crash on DNSSEC queries. Fixed in [commit 2194](#) and [commit 2196](#) (thanks to Charlie Smurthwaite) closing [ticket 360](#).
- Errors looking up a UID or GID were reported confusingly ('Success'), fixed in [commit 2195](#), closing [ticket 359](#).
- Fix compilation against older MySQL, client libraries ([commit 2198](#), [commit 2199](#), [commit 2204](#)), especially for older RHEL/CentOS. Also addresses the failure to look in lib64 directory for PostgreSQL.
- Sqlite3 needs write access not just to its database file, but also to the directory it is in. If this wasn't the case, no useful error message was provided. Improvement in [commit 2202](#).
- Update of MongoDB backend ([commit 2203](#), [commit 2212](#)).
- 'pdnssec hash-zone-record' emitted an inverted warning about narrow NSEC3 hashes. Spotted by Jan-Piet Mens, fix in [commit 2205](#).
- PowerDNS can fill out default fields for SOA records, but neglected to do so if the SOA record was matched by an incoming ANY question. Spotted by Marc Laros & others. Fixes [ticket 357](#), code in [commit 2206](#).
- PowerDNS would mistreat binary data in TXT records. Fix in [commit 2207](#). Again spotted by Jan-Piet Mens. Closes [ticket 356](#).
- Add experimental Lua backend by our star contributor Fredrik Danerklint. [commit 2208](#).
- Christoph Meerwald discovered our RRSIG freshness checking checked more than the intended RRSIG (on the SOA record). Fix in [commit 2209](#).
- Christoph Meerwald discovered we got confused by TSIG signed EDNS-adorned queries, since we expected the EDNS OPT pseudorecord to be the very last record. Fix in [commit 2214](#).
- Christoph Meerwald discovered that when using SOA outgoing editing we would sign and THEN edit. This was not productive. Fixed in [commit 2215](#).
- Add missing-but-documented pdnssec command 'disable-dnssec'. Spotted by Craig Whitmore. Plus fixed misleading --help output. Code in [commit 2216](#).
- By popular demand, a tweak which makes an overloaded database no longer restart PowerDNS but to drop queries until the database is available again. Code in [commit 2217](#), lightly tested. Enable by setting 'overload-queue-length=100' (for example).
- By suggestion of Miek Gieben of SIDN, add SOA-EDIT mode 'EPOCH' which sets the SOA serial number to the 'UNIX time'. Implemented in [commit 2218](#).
- Added some US export control & ECCN to documentation, needed because of DNSSEC content. Update in [commit 2219](#).
- Fix up various spelling mistakes and badly formatted messages ([commit 2220](#) and [commit 2221](#)) by Maik Zumstrull and 'anonymous'.
- After a lot of thought, we now handle CNAMEs to names outside our knowledge ('bailiwick') exactly as in BIND 9.8.0, even though our way was standards compliant too. It confused things. Update in [commit 2222](#) and [commit 2224](#).
- Tweak sqlite3 library location detection for newer Ubuntu versions. Change in [commit 2223](#).

- DNSSEC SQL schema improvements allowing for the use of constraints and foreign keys in [commit 2225](#), by Gerald Gruenberg, closing [ticket 371](#).
- Add support for EDNS option 'edns-subnet', based on draft-vandergaast-edns-client-subnet ([commit 2226](#), [commit 2228](#), [commit 2229](#), [commit 2230](#), [commit 2231](#), [commit 2233](#)).
- Silence SIGCHLD warning from Perl when used to power 'pipe' backends ([commit 2232](#)).
- Add experimental support, off by default, for draft-edns-subnet. See [commit 2233](#) and [commit 2239](#) for details how to use this feature.
- PostgreSQL and LDAP backends can now deal with a restart of their respective servers. Many thanks to Peter van Dijk for debugging and Nikolaos Milas for supplying a reproduction path of the problem (& much nagging). Fixes in [commit 2233](#) and [commit 2235](#).
- Jan-Piet Mens discovered that records inserted by Lua on zone retrieval did not get correct 'ordname' and 'auth' fields for DNSSEC. Fixed in [commit 2174](#).
- Silenced various relevant and less relevant compilation warnings ([commit 2175](#)). Thanks to Serge Belyshev for pointing out the error in our ways.
- Steve Bauer discovered we would cache empty recursive answers in some cases. Addressed in [commit 2176](#).
- James Cloos reported that 'pdnssec check-zone' tripped over SRV records. Fixed this, and added check-zone to the regression tests. Code in [commit 2177](#).
- DNSSEC regression tests were added in commits [2178](#), [2179](#), [2182](#), [2186](#) We test against the fine tools from NLNetLabs.
- Secure DNSSEC delegations to ourselves picked wrong zone to serve the DS record from. Fixed in [commit 2180](#), [commit 2181](#), [commit 2183](#). reported by Niek Willems of InterNLnet.
- Stef Van Dessel suggested we made our RPMs state explicitly that they need glibc 2.4 on Linux. Code in [commit 2184](#).
- John Leach discovered our MySQL based backends would wait for ages on a failing MySQL server. The patch merged in [commit 2189](#) reduces the timeout significantly, which is especially useful with haproxy and mysqlproxy.
- [commit 2190](#) fixes a crash reported by Marc Laros when using a non-DNSSEC capable backend. Should also improve non-DNSSEC performance.

Changes between RC1 and RC2:

- Zone2sql sent out the wrong 'COMMIT' statement in sqlite mode. In addition, in this mode, zone2sql would not emit statements to update the domains table unless the 'slave' setting was chosen. Code in [commit 2167](#).
- We dropped the Authoritative Answer flag on an out-of-bailiwick CNAME referral, which was unnecessary. Code in [commit 2170](#).
- Kees Monshouwer discovered that we failed to detect the location of PostgreSQL on RHEL/CentOS. Fix in [commit 2144](#). In addition, [commit 2162](#) eases detection of MySQL on RHEL/CentOS 64 bits systems.
- Marc Laros re-reported an old bug in the internally used 'pdns' backend where details of the SOA record were not filled out correctly. Resolved in [commit 2145](#).
- Jan-Piet Mens found that our TSIG signed SOA zone freshness check was signed incorrectly. Fixed in [commit 2147](#). Improved error messages that helped debug this issue in [commit 2148](#), [commit 2149](#).
- Jan-Piet Mens helped debug an issue where some servers were "almost always" unable to transfer a TSIG signed zone correctly. Turns out that the TSIG signing code used an internal timestamp and not the remote timestamp. Because of good NTP synchronization this quite often was not a problem. Fix in [commit 2159](#).
- Thor Spruyt of Telenet discovered that the PowerDNS code would try to emit DNS answers over TCP of over 65535 bytes long, which failed. We now truncate such answers properly. Code in [commit 2150](#).

- The Slave engine now reuses an existing database connection, removing the need to create a new database connection every minute (and worse, log about it). Code in [commit 2153](#).
- Fix a potential Year 2106 bug in the TSIG signing code. Because we care ([commit 2156](#)).
- Added experimental support for the 'DANE' TLSA record which is used to authenticate SSL certificates via DNSSEC. [commit 2161](#).
- Added experimental support for the MongoDB 'NoSQL' backend, contributed by Fredrik Danerklint in [commit 2162](#).

On to the release notes. Next to DNSSEC, other major new features include:

- TSIG for authorizing and authenticating AXFR requests & incoming zone transfers (Code in [2024](#), [2025](#), [2033](#), [2034](#)). This allows for retrieving TSIG protected content, as well as serving it.
- Per zone also-notify.
- MyDNS compatible backend, allowing for 'instantaneous' migration from this authoritative nameserver. Code in [commit 1418](#), contributed by Jonathan Oddy.
- PowerDNS can now slave zones over IPv6 and notify IPv6 remotes of updates. Already. Code in [commit 2009](#) and beyond.
- Lua based incoming zone editing, allowing masters or signing slaves to add information to the zone they will (re-)serve. Implemented in [commit 2065](#). To enable, use LUA-AXFR-SCRIPT zone metadata setting.
- Native Oracle backend with full DNSSEC support. Contributed by Maik Zumstrull, then at the Steinbuch Centre for Computing at the Karlsruhe Institute of Technology.
- "Also-notify" support, implemented by Aki Tuomi in [commit 1400](#). Support for Generic SQL backends and for the BIND backend. Further code in [commit 1360](#).
- Support for binding to thousands of IP addresses, code in [commit 1443](#).
- Generic MySQL backend now supports stored procedures. Implemented in [commit 2084](#), closing [ticket 231](#).
- Generic ODBC backend compiles again, and is reported to work for some users that need it. Code contributed in [ticket 309](#), author unknown.
- Massively parallel slaving infrastructure, able to check the freshness of thousands of remote zones per second, plus perform many incoming zone transfers simultaneously. Sponsored by Tyler Hall, code in [1449](#), [1500](#), [1859](#)
- Core DNS logic replaced completely to deal with the brave new world of DNSSEC.

Bugs fixed:

- sqlite2 and sqlite3 backends used MySQL-style escaping, leading to SQL errors in some cases. Discovered by Sten Spans. Fixed in [commit 1342](#).
- Internal webserver no longer prints '1e2%'. Bug rediscovered by Jeff Sipek. Fixed in [commit 1342](#).
- PowerDNS would refuse to serve domain names with spaces in them, or otherwise non-printable characters. Addressed in [commit 2081](#).
- PowerDNS can now serve escaped labels, as described by RFC 4343. Data should be present in backends in that escaped form. Code in [commit 2089](#).
- In some cases, we would include duplicate CNAMEs. In addition, we would hand out a full root-referral when not configured to in some cases ([ticket 223](#)). Discovered by Andreas Jakum, fixed in [commit 1344](#).
- Shane Kerr discovered we would corrupt DNS transaction IDs from the packet cache on big endian systems. Fix in [commit 1346](#), closing [ticket 222](#).
- PowerDNS did not use RFC 1982 serial arithmetic, leading to a SOA serial number of 1 to be regarded as older than 4400000000, when in fact it is 'newer'. Issue (re-)discovered by Jan-Piet Mens.

- BIND backend got confused of a zone's file name changed after a configuration reload. Fix in [commit 1347](#), closing [ticket 228](#).
- When restarted by the Guardian, PowerDNS will perform a full multi-threaded cache cleanup, which took a long time and could crash. Fix in [commit 1364](#).
- Under artificial circumstances, PowerDNS would never clean its packet cache. Found by Marcus Goller, fix in [commit 1399](#) and [commit 1408](#). This update also retunes the cleanup frequency.
- Packetcache would cache things it should not have been caching. Fixes in commits [1407](#), [1488](#), [1869](#), [1880](#)
- When processing incoming notifications, the BIND backend was case-sensitive, and would disregard notifications in the wrong case. Discovered by 'Dolphin', fix in [commit 1420](#).
- The init.d script did not mention the 'reload' command. Code in [commit 1463](#), closes [ticket 233](#).
- Generic SQL Backends would sometimes emit obscure error messages. Fix in [commit 2049](#).
- PowerDNS would be confused by embedded NULs in domain names, and would also mess up the escaping of some characters. Fix in [commit 1468](#), [commit 1469](#), [commit 1478](#), [commit 1480](#),
- SOA queries for the name of a delegation point were not referred. Fix in [commit 1466](#), closing [ticket 224](#). In addition, queries for AAAA for a CNAMEd record pointing to a name with no AAAA would deliver a direct SOA, without the CNAME in between. Fix in [commit 1542](#), [commit 1607](#). Also, wildcard CNAMEs pointing to a record without the type requested suffered from the same issue, fix in [commit 1543](#).
- On processing an incoming AXFR, once an MX or SRV record had been seen, all future fields got a 'priority' entry as well. This had no operational impact, but looked messy. Fixed in [commit 1437](#).
- Aki Tuomi discovered that the BIND zone file parser would misrepresent 'something IN MX 15 @'. Fix in [commit 1621](#).
- Marco Davids discovered the BIND zone file parser would trip over really long lines. Fix in [commit 1624](#), [commit 1625](#).
- Thomas Mieslinger discovered that our webserver would only be started after dropping privileges, which could cause problems. Fix in [commit 1629](#).
- Zone2sql did quite often not do exactly what was required, which users fixed by editing the SQL output. Revamped in [commit 2032](#).
- An Ubuntu user discovered in Launchpad bug 600479 that restarting database threads cost a lot of memory. Normally this is rare, except in case of problems. Addressed in [commit 1676](#).
- BIND backend could crash under (very) high load with very large numbers of zones (hundreds of thousands). Fixed in [commit 1690](#).
- Miek Gieben and Marco Davids spotted that PowerDNS would answer the version.bind query in the IN class too. Bug reported via twitter! Fix in [commit 1709](#).
- Marcus Lauer and the OpenDNSSEC project discovered that outgoing notifications did not carry the 'aa' flag. Fixed in [commit 1746](#).
- Debugging PowerDNS, or backgrounding it, could cause crashes. Fixed by Anders Kaseorg in [commit 1747](#).
- Fixed a bug that could cause crashes on launching thousands of backend connections. Never observed to occur, but who knows. Fix in [commit 1792](#).
- Under some circumstances, large answers could be truncated in mid-record. While technically legal, this upset a number of resolver implementations (including the PowerDNS Recursor!). Fixed in [commit 1830](#), re-closes [ticket 200](#).
- Jan Piet Mens and Florian Weimer discovered we had problems dealing with escaped labels and escaped TXT fields. Fixed in [commit 2000](#).
- After 2.2 billion queries, statistics would wrap oddly. Fix in [commit 2019](#), closing [ticket 327](#).

Improvements:

- Long TXT records are now split into 255-byte components automatically. Implemented in [commit 1340](#), reported by Darren Gamble in [ticket 188](#).
- When receiving large numbers of notifications, PowerDNS would check these synchronously, leading to a slowdown for other services. Fixed in [commit 2058](#), problem diagnosed by Richard Poole of Heart Internet.
- Fixed compilation on newer compilers and newer versions of Boost. Changes in [1345](#) (closes [ticket 227](#)), [1391](#), [1394](#), [1425](#), [1427](#), [1428](#), [1429](#), [1440](#), [1653](#), thanks to Ruben Kerkhof and others.
- Moved Generic PostgreSQL backend over to the newer E" style escapes. [commit 2094](#).
- Compilation fixes for Mac OS X 10.5.7 in [commit 1389](#), thanks to Tobias Markmann.
- We can now bind to scoped IPv6 addresses, lack spotted by Darren Gamble. Part of the fix is in [commit 2018](#).
- Built-in query cache can now also cache queries which lead to multiple answers. Code in [commit 2069](#).
- Prodded on by Jan Piet Mens, we now support 'unknown types' (which look like TYPE65534).
- Add 'slave-renotify' to retransmit notifies for slaved zones, which is helpful when acting as a 'signing slave' for a hidden master. Code in [commit 1950](#).
- No longer let zone2sql and zone2ldap import BIND 'hint' zones. [commit 1998](#).
- Allow for timestamps to explicitly be specified in (s)econds. Code in [commit 1398](#), closing [ticket 250](#).
- Zones with URL and MBOXFW records can be transferred over AXFR, code in [commit 1464](#).
- Maik Zumstrull cleaned up the BIND Backend makefile, plus taught our init.d script to read /etc/default/pdns. Code in [commit 1601](#), [commit 1602](#).
- Generic SQL backends now support multiple masters in the domains table. Code in [commit 1857](#). Additionally, masters can also have :port numbers. Code in [commit 1858](#).

1.3.6 Recursor version 3.3.1



Warning
Unreleased

Version 3.3.1 contains a small number of important fixes, adds some memory usage statistics, but no new features.

- Discovered by John J and Robin J, the PowerDNS Recursor did not process packets that were truncated in mid-record, and also did not act on the 'truncated' (TC) flag in that case. This broke a very small number of domains, most of them served by very old versions of the PowerDNS Authoritative Server. Fix in [commit 1740](#).
 - PowerDNS emitted a harmless, but irritating, error message on receiving certain very short packets. Discovered by Winfried A and John J, fix in [commit 1729](#).
 - PowerDNS could crash on startup if configured to provide service on malformed IPv6 addresses on FreeBSD, or in case when the FreeBSD kernel was compiled without any form of IPv6 support. Debugged by Bryan Seitz, fix in [commit 1727](#).
 - Add max-mthread-stack metric to debug rare crashes. Could be used to save memory on constrained systems. Implemented in [commit 1745](#).
 - Add cache-bytes and packetcache-bytes metrics to measure our 'pre-malloc' memory utilization. Implemented in [commit 1750](#).
-

1.3.7 Recursor version 3.3

**Warning**

Released on the 22nd of September 2010.

Version 3.3 fixes a number of small but persistent issues, rounds off our IPv6 %link-level support and adds an important feature for many users of the Lua scripts.

In addition, scalability on Solaris 10 is improved.

Bug fixes:

- 'dist-recursor' script was not compatible with pure POSIX /bin/sh, discovered by Simon Kirby. Fix in [commit 1545](#).
- Simon Bedford, Brad Dameron and Laurient Papier discovered relatively high TCP/IP loads could cause TCP/IP service to shut down over time. Addressed in commits [1546](#), [1640](#), [1652](#), [1685](#), [1698](#). Additional information provided by Zwane Mwaikambo, Nicholas Miell and Jeff Roberson. Testing by Christian Hofstaedtler and Michael Renner.
- The PowerDNS Recursor could not read the 'root zone' (this is something else than the root hints) because of an unquoted TXT record. This has now been addressed, allowing operators to hardcode the root zone. This can improve security if the root zone used is kept up to date. Change in [commit 1547](#).
- A return of an old bug, when a domain gets new nameservers, but the old nameservers continue to contain a copy of the domain, PowerDNS could get 'stuck' with the old servers. Fixed in [commit 1548](#).
- Discovered & reported by Alexander Gall of SWITCH, the Recursor used to try to resolve 'AXFR' records over UDP. Fix in [commit 1619](#).
- The Recursor embedded authoritative server messed up parsing a record like '@ IN MX 15 @'. Spotted by Aki Tuomi, fix in [commit 1621](#).
- The Recursor embedded authoritative server messed up parsing really really long lines. Spotted by Marco Davids, fix in [commit 1624](#), [commit 1625](#).
- Packet cache was not DNS class correct. Spotted by "Robin", fix in [commit 1688](#).
- The packet cache would cache some NXDOMAINs for too long. Solving this bug exposed an underlying oddity where the initial NXDOMAIN response had an overly long (untruncated) TTL, whereas all the next ones would be ok. Solved in [commit 1679](#), closing [ticket 281](#). Especially important for RBL operators. Fixed after some nagging by Alex Broens (thanks).

Improvements:

- The priming of the root now uses more IPv6 addresses. Change in [commit 1550](#), closes [ticket 287](#). Also, the IPv6 address of I.ROOT-SERVERS.NET was added in [commit 1650](#).
 - The `rec_control dump-cache` command now also dumps the 'negative query' cache. Code in [commit 1713](#).
 - PowerDNS Recursor can now bind to fe80 IPv6 space with '%eth0' link selection. Suggested by Darren Gamble, implemented with help from Niels Bakker. Change in [commit 1620](#).
 - Solaris on x86 has a long standing bug in `port_getn()`, which we now work around. Spotted by 'Dirk' and 'AS'. Solution suggested by the Apache runtime library, update in [commit 1622](#).
 - New runtime statistic: 'tcp-clients' which lists the number of currently active TCP/IP clients. Code in [commit 1623](#).
 - Deal better with UltraDNS style CNAME redirects containing SOA records. Spotted by Andy Fletcher from UKDedicated in [ticket 303](#), fix in [commit 1628](#).
-

- The packet cache, which has 'ready to use' packets containing answers, now artificially ages the ready to use packets. Code in [commit 1630](#).
- Lua scripts can now indicate that certain queries will have 'variable' answers, which means that the packet cache will not touch these answers. This is great for overriding some domains for some users, but not all of them. Use `setvariable()` in Lua to indicate such domains. Code in [commit 1636](#).
- Add query statistic called 'dont-outqueries', plus add IPv6 address `::` and IPv4 address `0.0.0.0` to the default "dont-query" set, preventing the Recursor from talking to itself. Code in [commit 1637](#).
- Work around a gcc 4.1 bug, still in wide use on common platforms. Code in [commit 1653](#).
- Add 'ARCHFLAGS' to PowerDNS Recursor Makefile, easing 64 bit compilation on mainly 32 bit platforms (and vice versa).
- Under rare circumstances, querying the Recursor for statistics under very high load could lead to a crash (although this has never been observed). Bad code removed & good code unified in [commit 1675](#).
- Spotted by Jeff Sipek, the `rec_control` manpage did not list the new `get-all` command. [commit 1677](#).
- On some platforms, it may be better to have PowerDNS itself distribute queries over threads (instead of leaving it up to the kernel). This experimental feature can be enabled with the '`pdns-distributes-queries`' setting. Code in [commit 1678](#) and beyond. Speeds up Solaris measurably.
- Cache cleaning code was cleaned up, unified and expanded to cover the 'negative cache', which used to be cleaned rather bluntly. Code in [commit 1702](#), further tweaks in [commit 1712](#), spotted by Darren Gamble, Imre Gergely and Christian Kovacic.

Changes between RC1, RC2 and RC3.

- RC2: Fixed linking on RHEL5/CentOS5, which both ship with a gcc compiler that claims to support atomic operations, but doesn't. Code in [commit 1714](#). Spotted by 'Bas' and Imre Gergely.
- RC2: Negative query cache was configured to grow too large, and was not cleaned efficiently. Code in [commit 1712](#), spotted by Imre Gergely.
- RC3: Root failed to be renewed automatically, relied on fallback to make this happen. Code in [commit 1716](#), spotted by Detlef Peeters.

1.3.8 Recursor version 3.2



Warning

Released on the 7th of March 2010.



Warning

Lua scripts from version 3.1.7.* are fully compatible with version 3.2. However, scripts written for development snapshot releases, are NOT. Please see Section [17.7](#) for details!

The 3.2 release is the first major release of the PowerDNS Recursor in a long time. Partly this is because 3.1.7.* functioned very well, and delivered satisfying performance, partly this is because in order to really move forward, some heavy lifting had to be done.

As always, we are grateful for the large PowerDNS community that is actively involved in improving the quality of our software, be it by submitting patches, by testing development versions of our software or helping debug interesting issues. We specifically want to thank Stefan Schmidt and Florian Weimer, who both over the years have helped tremendously in keeping PowerDNS fast, stable and secure.

This version of the PowerDNS Recursor contains a rather novel form of lock-free multithreading, a situation that comes close to the old '--fork' trick, but allows the Recursor to fully utilize multiple CPUs, while delivering unified statistics and operational control.

In effect, this delivers the best of both worlds: near linear scaling, with almost no administrative overhead.

Compared to 'regular multithreading', whereby threads cooperate more closely, more memory is used, since each thread maintains its own DNS cache. However, given the economics, and the relatively limited total amount of memory needed for high performance, this price is well worth it.

In practical numbers, over 40,000 queries/second sustained performance has now been measured by a third party, with a 100.0% packet response rate. This means that the needs of around 400,000 residential connections can now be met by a single commodity server.

In addition to the above, the PowerDNS Recursor is now providing resolver service for many more Internet users than ever before. This has brought with it 24/7 Service Level Agreements, and 24/7 operational monitoring by networking personnel at some of the largest telecommunications companies in the world.

In order to facilitate such operation, more statistics are now provided that allow the visual verification of proper PowerDNS Recursor operation. As an example of this there are now graphs that plot how many queries were dropped by the operating system because of a CPU overload, plus statistics that can be monitored to determine if the PowerDNS deployment is under a spoofing attack.

All in all, this is a large and important PowerDNS Release, paving the way for further innovation.

Note

This release removes support for the 'fork' multi-processor option. In addition, the default is now to spawn two threads. This has been done in such a way that total memory usage will remain identical, so each thread will use half of the allocated maximum number of cache entries.

Changes between RC2 and -release:

- 'Make install' when an existing configuration file contained a 'fork' statement has been fixed. Spotted by Darren Gamble, code in [commit 1534](#).
- Reloading a non-existent allow-from-file caused the control thread to stop working. Spotted by Imre Gergely, code in [commit 1532](#).
- Parser got confused by reading an empty line in auth-forward-zones. Spotted by Imre Gergely, code in [commit 1533](#).
- David Gavarret discovered undocumented and not-working settings to set the owner, group and access modes of the control socket. Code by Aki Tuomi and documentation in [commit 1535](#). Fixup in [commit 1536](#) for FreeBSD as found by Ralf van der Enden.
- Tiny improvement possibly solving an issue on Solaris 10's completion port event multiplexer ([commit 1537](#)).

Changes between RC1 and RC2:

- Compilation on Solaris 10 has been fixed (various patchlevels had different issues), code in [commit 1522](#).
 - Compatibility with CentOS4/RHEL4 has been restored, the gcc and glibc versions shipped with this distribution contain a Thread Local Storage bug which we now work around. Thanks to Darren Gamble and Imre Gergely for debugging this issue, code in [commit 1527](#).
 - A failed setuid operation, because of misconfiguration, would result in a crash instead of an error message. Fixed in [commit 1523](#).
 - Imre Gergely discovered that PowerDNS was doing spurious root repriming when invalidating nssets. Fixed in [commit 1531](#).
 - Imre Gergely discovered our rrd graphs had not been changed for the new multithreaded world, and did not allow scaling beyond 200% cpu use. In addition, CPU usage graphs did not add up correctly. Implemented in [commit 1524](#).
-

- Andreas Jakum discovered the description of 'max-packetcache-entries' and 'forward-zones-recurse' was wrong in the output of '--help' and '--config'. In addition, some stray backup files made it into the RC1 release. Addressed in [commit 1529](#).

Full release notes follow, including some overlap with the incremental release notes above. Improvements:

- Multithreading, allowing near linear scaling to multiple CPUs or cores. Configured using 'threads=' (many commits). This also deprecates the '--fork' option.
- Added ability to read a configuration item of a running PowerDNS Recursor using 'rec_control get-parameter' ([commit 1243](#)), suggested by Wouter de Jong.
- Added ability to read all statistics in one go of a running PowerDNS Recursor using 'rec_control get-all' ([commit 1496](#)), suggested by Michael Renner.
- Speedups in packet generation (Commits [1258](#), [1259](#), [1262](#))
- TCP deferred accept() filter is turned on again for slight DoS protection. Code in [commit 1414](#).
- PowerDNS Recursor can now do TCP/IP queries to remote IPv6 addresses ([commit 1412](#)).
- Solaris 9 '/dev/poll' support added, Solaris 8 now deprecated. Changes in [commit 1421](#), [commit 1422](#), [commit 1424](#), [commit 1413](#).
- Lua functions can now also see the address `_to_` which a question was sent, using `getlocaladdress()`. Implemented in [commit 1309](#) and [commit 1315](#).
- Maximum cache sizes now default to a sensible value. Suggested by Roel van der Made, implemented in [commit 1354](#).
- Domains can now be forwarded to IPv6 addresses too, using either `::1` syntax or `::1]:25`. Thanks to Wijnand Modderman for discovering this issue, fixed in [commit 1349](#).
- Lua scripts can now load libraries at runtime, for example to calculate md5 hashes. Code by Winfried Angele in [commit 1405](#).
- Periodic statistics output now includes average queries per second, as well as packet cache numbers ([commit 1493](#)).
- New metrics are available for graphing, plus added to the default graphs ([commit 1495](#), [commit 1498](#), [commit 1503](#))
- Fix errors/crashes on more recent versions of Solaris 10, where the ports functions could return ENOENT under some circumstances. Reported and debugged by Jan Gyselinck, fixed in [commit 1372](#).

New features:

- Add `pdnslog()` function for Lua scripts, so errors or other messages can be logged properly.
- New settings to set the owner, group and access modes of the control socket (`socket-owner`, `socket-group`, `socket-mode`). Code by Aki Tuomi and documentation in [commit 1535](#). Fixup in [commit 1536](#) for FreeBSD as found by Ralf van der Enden.
- `rec_control` now accepts a `--timeout` parameter, which can be useful when reloading huge Lua scripts. Implemented in [commit 1366](#).
- Domains can now be forwarded with the 'recursion-desired' bit on or off, using either **forward-zones-recurse** or by prefixing the name of a zone with a '+' in **forward-zones-file**. Feature suggested by Darren Gamble, implemented in [commit 1451](#).
- Access control lists can now be reloaded at runtime (implemented in [commit 1457](#)).
- PowerDNS Recursor can now use a pool of query-local-addresses to further increase resilience against spoofing. Suggested by Ad Spelt, implemented in [commit 1426](#).
- PowerDNS Recursor now also has a packet cache, greatly speeding up operations. Implemented in [commit 1426](#), [commit 1433](#) and further.
- Cache can be limited in how long it maximally stores records, for BIND compatibility (TTL limiting), by setting **max-cache-ttl**. Idea by Winfried Angele, implemented in [commit 1438](#).

- Cache cleaning turned out to be scanning more of the cache than necessary for cache maintenance. In addition, far more frequent but smaller cache cleanups improve responsiveness. Thanks to Winfried Angele for discovering this issue. (commits [1501](#), [1507](#))
- Performance graphs enhanced with separate CPU load and cache effectiveness plots, plus display of various overload situations (commits [1503](#))

Compiler/Operating system/Library updates:

- PowerDNS Recursor can now compile against newer versions of Boost (verified up to and including 1.42.0). Reported & fixed by Darix in [commit 1274](#). Further fixes in [commit 1275](#), [commit 1276](#), [commit 1277](#), [commit 1283](#).
- Fix compatibility with newer versions of GCC (closes ticket [ticket 227](#), spotted by Ruben Kerkhof, code in [commit 1345](#), more fixes in [commit 1394](#), [1416](#), [1440](#)).
- Rrdtool update graph is now compatible with FreeBSD out of the box. Thanks to Bryan Seitz ([commit 1517](#)).
- Fix up Makefile for older versions of Make ([commit 1229](#)).
- Solaris compilation improvements (out of the box, no handwork needed).
- Solaris 9 MTasker compilation fixes, as suggested by John Levon. Changes in [commit 1431](#).

Bug fixes:

- Under rare circumstances, the recursor could crash on 64 bit Linux systems running glibc 2.7, as found in Debian Lenny. These circumstances became a lot less rare for the 3.2 release. Discovered by Andreas Jakum and debugged by #powerdns, fix in [commit 1519](#).
- Imre Gergely discovered that PowerDNS was doing spurious root repriming when invalidating nssets. Fixed in [commit 1531](#).
- Configuration parser is now resistant against trailing tabs and other whitespace ([commit 1242](#))
- Fix typo in a Lua error message. Close [ticket 210](#), as reported by Stefan Schmidt ([commit 1319](#)).
- Profiled-build instructions were broken, discovered & fixes suggested by Stefan Schmidt. [ticket 239](#), fix in [commit 1462](#).
- Fix up duplicate SOA from a remote authoritative server from showing up in our output ([commit 1475](#)).
- All security fixes from 3.1.7.2 are included.
- Under highly exceptional circumstances on FreeBSD the PowerDNS Recursor could crash because of a TCP/IP error. Reported and fixed by Andrei Poelov in [ticket 192](#), fixed in [commit 1280](#).
- PowerDNS Recursor can be a root-server again. Error spotted by the ever vigilant Darren Gamble (t229), fix in [commit 1458](#).
- Rare TCP/IP errors no longer lead to PowerDNS Recursor logging errors or becoming confused. Debugged by Josh Berry of Plusnet PLC. Code in [commit 1457](#).
- Do not hammer parent servers in case child zones are misconfigured, requery at most once every 10 seconds. Reported & investigated by Stefan Schmidt and Andreas Jakum, fixed in [commit 1265](#).
- Properly process answers from remote authoritative servers that send error answers without including the original question ([commit 1329](#), [commit 1327](#)).
- No longer spontaneously turn on 'export-etc-hosts' after reloading zones. Discovered by Paul Cairney, reported in [ticket 225](#), addressed in [commit 1348](#).
- Very abrupt server failure of large numbers of high-volume authoritative servers could trigger an out of memory situation. Addressed in [commit 1505](#).
- Make timeouts for queries to remote authoritative servers configurable with millisecond granularity. In addition, the old code turned out to consider the timeout expired when the integral number of seconds since 1970 increased by 1 - which *on average* is after 500ms. This might have caused spurious timeouts! New default timeout is 1500ms. See **network-timeout** setting for more details. Code in [commit 1402](#).

1.3.9 Recursor version 3.1.7.2

**Warning**

Released on the 6th of January 2010.

This release consist of a number of vital security updates. These updates address issues that can in all likelihood lead to a full system compromise. In addition, it is possible for third parties to pollute your cache with dangerous data, exposing your users to possible harm.

This version has been well tested, and at the time of this release is already powering millions of internet connections, and should therefore be a risk-free upgrade from 3.1.7.1 or any earlier version of the PowerDNS Recursor.

All known versions of the PowerDNS Recursor are impacted to a greater or lesser extent, so an immediate update is advised.

These vulnerabilities were discovered by a third party that can't yet be named, but who we thank for their contribution to a more secure PowerDNS Recursor.

For more information, see Section [1.10](#) and Section [1.11](#).

1.3.10 Recursor version 3.1.7.1

**Warning**

Released on the 2nd of August 2009.

This release consists entirely of fixes for tiny bugs that have been reported over the past year. In addition, compatibility has been restored with the latest versions of the gcc compiler and the 'boost' libraries.

No features have been added, but some debugging code that very slightly impacted performance (and polluted the console when operating in the foreground) has been removed.

FreeBSD users may want to upgrade because of a very remote chance of 3.1.7 and previous crashing once every few years. For other operators not currently experiencing problems, there is no reason to upgrade.

- Improved error messages when parsing zones for authoritative serving ([commit 1235](#)).
 - Better resilience against whitespace in configuration (changesets [1237](#), [1240](#), [1242](#))
 - Slight performance increase ([commit 1378](#))
 - Fix rare case where timeouts were not being reported to the right query-thread ([commit 1260](#))
 - Fix compilation against newer versions of the Boost C++ libraries ([commit 1381](#))
 - Close very rare issue with TCP/IP close reporting ECONNRESET on FreeBSD. Reported by Andrei Poelov in [ticket 192](#).
 - Silence debugging output ([commit 1286](#)).
 - Fix compilation against newer versions of gcc ([commit 1384](#))
 - No longer set export-etc-hosts to 'on' on reload-zones. Discovered by Paul Cairney, closes [ticket 225](#).
 - Sane default for the maximum cache size in the Recursor, suggested by Roel van der Made ([commit 1354](#)).
 - No longer exit because of the changed behaviour of the Solaris 'completion ports' in more recent versions of Solaris. Fix in [commit 1372](#), reported by Jan Gyselink.
-

1.3.11 Authoritative Server version 2.9.22

**Warning**

Released on the 27th of January 2009.

This is a huge release, spanning almost 20 months of development. Besides fixing a lot of bugs, of note is the addition of the so called 'Notification Proxy', which allows PowerDNS to function as a master server behind a firewall, plus the huge performance improvement of the internal caches.

This work has been made possible by UPC Broadband and Directi, respectively.

Finally, the release candidates of this version have been tested & improved by Jorn Ekkelenkamp, Ton van Rosmalen, Jeff Sipek, Tyler Hall, Christof Meerwald and Stefan Schmidt.

Fixed between rc1 and rc2, but not an issue in 2.9.21.

- **pdns_control counts** again outputs proper cache statistics. Implemented in [commit 1304](#).
- Negative query caching was reinstated, leading to 6 times fewer backend queries than rc1 on the Express.powerdns.com servers.
- Packetcache no longer needlessly parses outgoing packets before sending them.
- Fancy records work again. This work has been sponsored by ISP Services. Implemented in [commit 1302](#) and [commit 1299](#).

New features:

- **pdns_control** can now also work over TCP/IP. Sponsored by Directi. Commits [1246](#), [1251](#), [1254](#), [1255](#).
- Implemented a notification proxy, see Section [24.1](#). This work was sponsored by UPC Broadband. Implemented in commits [1075](#), [1077](#), [1082](#), [1083](#), [1085](#) and [1086](#).
- IXFR queries are now supported in the sense that we treat them as AXFR queries, silencing warnings in other nameservers. Suggested in [ticket 131](#).
- The PIPE backend has been extended by David Apgar to allow the reporting of errors using the 'FAIL' command, plus support for responses with whitespace. Implemented in [commit 1114](#).
- PowerDNS Authoritative server now parses incoming EDNS options, like maximum allowed packet size. Implemented in [commit 1123](#) and [commit 1281](#).
- Added support for DHCID, IPSECKEY and KX records, thanks Norbert Sendetzky for the hint. Implemented in [commit 1144](#).
- Norbert Sendetzky has added support for all record types supported by PowerDNS to the LDAPBackend. Furthermore, the detection of OpenLDAP in autoconf has been improved. Finally, debian has supplied some fixes to PowerLDAP. Implemented in [commit 1152](#) and [commit 1153](#).
- Implemented EDNS NSID option for retrieving the nameserver ID out of band. Defaults to hostname, can be specified using the **server-id** setting. Code in [commit 1232](#).
- Implemented experimental EDNS PING for enhanced forgery resilience. Code in [commit 1232](#).

Performance:

- Improve packet generation performance, in some cases by 25%. Code in [1258](#), [1259](#).
 - Improved access list checking performance. [commit 1261](#).
-

- PowerDNS Authoritative caches were completely redone, and are now based on the same cache that is in the resolver. This work has been sponsored by Directi. In large benchmarks, PowerDNS performance has improved by an order of magnitude or more. This new version allows for near-instantaneous cache purging, plus very rapid purging based on suffix. Purge commands can also be batched. This work is partially based on an innovative reverse-string comparison function authored by Aki Tuomi.
- Installations which run with very high cache hitrates can now benefit from multiple CPUs by setting **receiver-threads** to the number of desired CPUs to utilize in cache operations. Implemented in [commit 1316](#).
- BIND backend speedups in [commit 1108](#), measured at around a 20% improvement, possibly more on very large setups.

Bugs fixed:

- Tyler Hall discovered the PowerDNS configuration file parser had problems with trailing tabs. This turned out to be a wider problem in PowerDNS. Buggy code replaced by a library call in [commit 1237](#) and [commit 1240](#).
- David Apgar of Yahoo discovered that our 'guardian' method of restarting PowerDNS in case of problems was not fool proof, and submitted a fix. A variation of this fix can be found in [commit 1323](#). Also reported by Directi.
- Connection reset by peer events in the TCP nameserver no longer lead to the cycling of database connections. Code in [commit 1241](#).
- FreeBSD compilation with Generic PostgreSQL backend was fixed. Reported by Wouter de Jong of WideXS, fixed in [commit 1305](#), closes [ticket 95](#).
- Webserver no longer prints '1e2%'. Finally closes [ticket 26](#). Much friendly nagging for over 3 years by Jeff Sipek, code in [commit 1303](#).
- PowerDNS used to ignore certain queries it could not answer. These queries are no longer ignored, but get a SERVFAIL response. Implemented in [commit 1239](#).
- Fix subtle CNAME and wildcard interactions reported by 'zzyzz', implemented in [commit 1147](#).
- The generic backends did not honour the **default-ttl** setting. Spotted and implemented by Matti Hiljanen.
- Matti Hiljanen discovered that the OpenDBX backend did not fill out the SOA ttl value properly. Matti also improved the SQL statements for better compatibility. Implemented in [commit 1181](#).
- Treat invalid WWW requests better. Spotted by Maikel Verheijen, implemented in [commit 1092](#).
- Documentation errors and typos, spotted by Marco Davids ([commit 1097](#)) and Rejo Zengers ([commit 1119](#))
- Properly fill out the 'recursion available'-flag. Spotted by Augie Schwer in [ticket 167](#).
- Several memory leaks on bad data in the database or other errors have been fixed. Addressed in [1078](#) and [1079](#).
- In contravention to the documentation, the domain type as specified in the database ('MASTER', 'SLAVE' or 'NATIVE') was interpreted case sensitively. [1084](#).
- BIND backend could crash on processing information about slave zones to be checked. Spotted by Stefan Schmidt, fixed in [1089](#).
- Jelte Jansen of Stichting NLNetLabs discovered PowerDNS in BIND mode couldn't operate as a root-server! Fixed in [1057](#).
- 'DPS' discovered there was a rare opportunity for PowerDNS to lock up waiting for new data. Addressed in [1076](#).
- Make singlethreaded mode more resilient against errors. [commit 1272](#).
- DNSSEC records were part of 2.9.21, but were not actually hooked up. Please note that while PowerDNS can serve most DNSSEC records, it does not do DNSSEC processing. Implemented in [1046](#).
- Shawn Starr migrated all his domains to PowerDNS in one evening, from an installation that had been used since BIND4. In doing so, he found 3 bugs in as many hours. An **IN** statement in the BIND `named.conf` with a zone with a trailing dot was misparsed, fixed in [commit 1233](#). Secondly, the zone file parser tripped over a line consisting of nothing but comments in the wrong place. Finally '\$ORIGIN .' was misparsed. Last two issues fixed in [commit 1234](#).

- Our statistics counters did not wrap correctly after the 2.15 billion mark. Spotted by Stefan Schmidt, reported in [ticket 179](#), fixed in [commit 1284](#).
- Bindbackend could sometimes generate very strange error messages while processing a malformed zone file. Sometimes such error messages could cause a crash (reported on HP-UX). Addressed by [commit 1279](#). This could not be triggered remotely. Closes ticket [ticket 203](#).
- Pipe backend did not clean up killed coprocesses. Found and fixed by Daniel Drown
- Installations with tens of thousands of slave domains would never complete the cycle to check the freshness of all zones as each incoming notification disrupted this cycle. Addressed in cooperation with Tyler Hall of EditDNS.

Improvements:

- Zone parser improvements mean \$TTL and \$INCLUDES now work a lot better. Implemented in [1056](#), [1062](#).
- No longer report temporary recvfrom errors, which used to spam the log on many systems. Addressed in [commit 1320](#).
- Direct queries for 'fancy records' would lead to errors, such queries now fail early. Spotted by Jorn Ekkelenkamp, implemented in [1051](#).
- Fix typo in geobackend, closing [ticket 157](#), implemented in [1090](#).
- Initial work on TSIG support - not done yet. Spurred on by Marco Davids.
- Embarrassingly, the 'master' configuration setting was not documented in the list of all settings!
- Norbert has updated OpenDBX so that SQLite reads and writes no longer deadlock, plus compilation fixes on Solaris, plus the addition of autoserials to backends that support triggers. Implemented in [commit 1154](#).
- Random generator is now based on AES, improving the security of certain proxy operations. This is the same random generator that is in the recursor. Implemented in [commit 1256](#).
- Documentation for 'supermaster' mode was improved due to popular demand.
- When binding to a UDP port failed, supply a more precise error message ([commit 1245](#))
- The zone parser error messages were vastly improved, partially inspired by Shawn's cowboy migration. Code in [commit 1235](#).
- Labels are compressed more efficiently (case-insensitively), leading to smaller packets. Implemented in [commit 1156](#).
- Fix handling of TCP timeouts to not cause a reload of the backends. Implemented in [commit 1092](#).
- TCP Receiver no longer spams the log with common network errors. Implemented in [commit 1306](#).
- Move from select() to poll()-based multiplexing, allowing PowerDNS to listen on more than 1024 sockets simultaneously. One big PowerDNS user needs this. Implemented in [1072](#).
- Zone2sql now reads source files in performance enhancing inode order. Additionally, zone2sql no longer dies on a missing zone file if **--on-error-resume-next** was specified. Finally, statistics of zone2sql conversion have been improved. Implemented in [1055](#).
- Address issues found by more recent g++ versions. Spotted and/or fixed by Jorn Ekkelenkamp ([commit 1051](#)), Marcus Rueckert ([commit 1094](#)), Norbert Sendetzky ([commit 1107](#)), Serge Belyshev ([commit 1171](#)).
- The Intel C Compiler implements certain things differently, causing the master/slave communicator to malfunction. Spotted by Marcus Rueckert, implemented in [1052](#), plus fallout in [1105](#).
- PowerDNS can now be compiled with Boost 1.37.0.
- Andre Lorbach of Adiscon discovered the Microsoft Windows 2003 nameserver adds out of zone data to zone transfers, which we need to ignore, instead of rejecting the entire zone. Implemented in [1048](#).
- PowerDNS now skips remote master servers which consistently generate timeout messages, improving the master checking cycle time tremendously. Developed in cooperation with Tyler Hall. Implemented in [commit 1278](#).

- When binding to a UDP port failed, supply a more precise error message ([commit 1245](#))
- **dnsreplay** now waits for the final answers to arrive, making it possible to process even small pcap files and get meaningful statistics. [commit 1268](#).
- **dnsreplay** has a more sane default timeout now, which can be configured too. Suggested by Augie Schwer in [ticket 163](#), implemented in [commit 1287](#).

1.3.12 Authoritative Server version 2.9.21.2

Released on the 18th of November 2008.

This release consists of a single patch to PowerDNS Authoritative Server version 2.9.21.1. In some configurations, notably with configuration option 'distributor-threads=1', the PowerDNS Authoritative Server crashes easily in some error conditions.

All users are urged to upgrade. Even though PowerDNS restarts itself on encountering such error conditions, and even though most PowerDNS configurations do not run in single threaded mode, an upgrade is recommended.

More detail can be found in Section [1.9](#).

1.3.13 Authoritative Server version 2.9.21.1

Released on the 6th of August 2008.

This release consists of a single patch to PowerDNS Authoritative Server version 2.9.21. Brian J. Dowling of Simplicity Communications has discovered a security implication of the previous PowerDNS behaviour to drop queries it considers malformed. We are grateful that Brian notified us quickly about this problem.

This issue has been assigned CVE-2008-3337. The single patch is in [commit 1239](#). More detail can be found in Section [1.8](#).

The implication is that while the PowerDNS Authoritative server itself does not face a security risk because of dropping these malformed queries, other resolving nameservers run a higher risk of accepting spoofed answers for domains being hosted by PowerDNS Authoritative Servers before 2.9.21.1.

While the dropping of queries does not aid sophisticated spoofing attempts, it does facilitate simpler attacks.

It may be good to know that several large sites already run with this patch applied, as it has been in the public code base for some weeks already.

1.3.14 Recursor version 3.1.7

Released the 25th of June 2008.

This version contains powerful scripting abilities, allowing operators to modify DNS responses in many interesting ways. Among other things, these abilities can be used to filter out malware domains, to perform load balancing, to comply with legal and other requirements and finally, to implement 'NXDOMAIN' redirection.

It is hoped that the addition of Lua scripting will enable responsible DNS modification for those that need it.

For more details about the Lua scripting, which can be modified, loaded and unloaded at runtime, see Section [17.7](#). Many thanks are due to the #lua irc channel, for excellent near-realtime Lua support. In addition, a number of PowerDNS users have been enthusiastically testing prereleases of the scripting support, and have found and solved many issues.

In addition, 3.1.7 fixes a number of bugs:

- In 3.1.5 and 3.1.6, an authoritative server could continue to renew its authority, even though a domain had been delegated to other servers in the meantime.

In the rare cases where this happened, and the old servers were not shut down, the observed effect is that users were fed outdated data.

Bug spotted and analysed by Darren Gamble, fix in [commit 1182](#) and [commit 1183](#).

- Thanks to long time PowerDNS contributor Stefan Arentz, for the first time, Mac OS X 10.5 users can compile and run the PowerDNS Recursor! Patch in [commit 1185](#).
- Sten Spans spotted that for outgoing TCP/IP queries, the **query-local-address** setting was not honored. Fixed in [commit 1190](#).
- **rec_control wipe-cache** now also wipes domains from the negative cache, hurrying up the expiry of negatively cached records. Suggested by Simon Kirby, implemented in [commit 1204](#).
- When a forwarder server is configured for a domain, using the **forward-zones** setting, this server IP address was filtered using the **dont-query** setting, which is generally not what is desired: the server to which queries are forwarded will often live in private IP space, and the operator should be trusted to know what he is doing. Reported and argued by Simon Kirby, fix in [commit 1211](#).
- Marcus Rueckert of OpenSUSE reported that very recent gcc versions emitted a (correct) warning on an overly complicated line in syncres.cc, fixed in [commit 1189](#).
- Stefan Schmidt discovered that the netmask matching code, used by the new Lua scripts, but also by all other parts of PowerDNS, had problems with explicit '/32' matches. Fixed in [commit 1205](#).

1.3.15 Recursor version 3.1.6

Released on the 1st of May 2008.

This version fixes two important problems, each on its own important enough to justify a quick upgrade.

- Version 3.1.5 had problems resolving several slightly misconfigured domains, including for a time 'juniper.net'. Nameserver timeouts were not being processed correctly, leading PowerDNS to not update the internal clock, which in turn meant that any queries immediately following an error would time out as well. Because of retries, this would usually not be a problem except on very busy servers, for domains with different nameservers at different levels of the DNS-hierarchy, like 'juniper.net'. This issue was fixed rapidly because of the help of [XS4ALL](#) (Eric Veldhuyzen, Kai Storbeck), Brad Dameron and Kees Monshouwer. Fix in [commit 1178](#).
- The new high-quality random generator was not used for all random numbers, especially in source port selection. This means that 3.1.5 is still a lot more secure than 3.1.4 was, and its algorithms more secure than most other nameservers, but it also means 3.1.5 is not as secure as it could be. A quick upgrade is recommended. Discovered by Thomas Biege of Novell (SUSE), fixed in [commit 1179](#).

1.3.16 Recursor version 3.1.5

Released on the 31st of March 2008.

Much like 3.1.4, this release does not add a lot of major features. Instead, performance has been improved significantly (estimated at around 20%), and many rare and not so rare issues were addressed. Multi-part TXT records now work as expected - the only significant functional bug found in 15 months. One of the oldest feature requests was fulfilled: version 3.1.5 can finally forward queries for designated domains to multiple servers, on differing port numbers if needed. Previously only one forwarder address was supported. This lack held back a number of migrations to PowerDNS.

We would like to thank Amit Klein of Trusteer for bringing a serious vulnerability to our attention which would enable a smart attacker to 'spooof' previous versions of the PowerDNS Recursor into accepting possibly malicious data.

Details can be found on [this Trusteer page](#).

It is recommended that all users of the PowerDNS Recursor upgrade to 3.1.5 as soon as practicable, while we simultaneously note that busy servers are less susceptible to the attack, but not immune.

The PowerDNS Security Advisory can be found in [Section 1.7](#).

This version can properly benefit from all IPv4 and IPv6 addresses in use at the root-servers as of early February 2008. In order to implement this, changes were made to how the Recursor deals internally with A and AAAA queries for nameservers, see below for more details.

Additionally, newer releases of the G++ compiler required some fixes (see [ticket 173](#)).

This release was made possible by the help of Wichert Akkerman, Winfried Angele, Arnoud Bakker (Fox-IT), Niels Bakker (no relation!), Leo Baltus (Nederlandse Publieke Omroep), Marco Davids (SIDN), David Gavarret (Neuf Cegetel), Peter Gervai, Marcus Goller (UPC), Matti Hiljanen (Saunalahti/Elisa), Ruben Kerkhof, Alex Kiernan, Amit Klein (Trusteer), Kenneth Marshall (Rice University), Thomas Rietz, Marcus Rueckert (OpenSUSE), Augie Schwer (Sonix), Sten Spans (Bit), Stefan Schmidt (Freenet), Kai Storbeck (xs4all), Alex Trull, Andrew Turnbull (No Wires) and Aaron Thompson, and many more who filed bugs anonymously, or who we forgot to mention.

Security related issues:

- Amit Klein has informed us that System random generator output can be predicted based on its past behaviour, allowing a smart attacker to 'spoof' our nameserver. Full details in [Section 1.7](#).
- The Recursor will by default no longer query private-space nameservers. This closes a slight security risk and simultaneously improves performance and stability. For more information, see **dont-query** in [Section 17.1](#). Implemented in [commit 923](#).
- Applied fix for [ticket 110](#) ('PowerDNS should change directory to '/' in chroot), implemented in [commit 944](#).

Performance:

- The DNS packet writing and parsing infrastructure performance was improved in several ways, see commits [925](#), [926](#), [928](#), [931](#), [1021](#), [1050](#).
- Remove multithreading overhead from the Recursor ([commit 999](#)).

Bug fixes:

- Built-in authoritative server now properly derives the TTL from the SOA record if not specified. Implemented in [commit 1165](#). Additionally, even when TTL was specified for the built-in authoritative server, it was ignored. Reported by Stefan Schmidt, closing [ticket 147](#).
- Empty TXT record components can now be served. Implemented in [commit 1166](#), closing [ticket 178](#). Spotted by Matti Hiljanen.
- The Recursor would not properly override old data with new, sometimes serving old and new data concurrently. Fixed in [commit 1137](#).
- SOA records with embedded carriage-return characters are now parsed correctly. Implemented in [commit 1167](#), closing [ticket 162](#).
- Some routing conditions could cause UDP connected sockets to generate an error which PowerDNS did not deal with properly, leading to a leaked file descriptor. As these run out over time, the recursor could crash. This would also happen for IPv6 queries on a host with no IPv6 connectivity. Thanks to Kai of xs4all and Wichert Akkerman for reporting this issue. Fix in [commit 1133](#).
- Empty unknown record types can now be stored without generating a scary error ([commit 1129](#))
- Applied fix for [ticket 111](#), [ticket 112](#) and [ticket 153](#) - large (multipart) TXT records are now retrieved and served properly. Fix in [commit 996](#).
- Solaris compilation instructions in Recursor documentation were wrong, leading to an instant crash on startup. Luckily nobody reads the documentation, except for Marcus Goller who found the error. Fixed in [commit 1124](#).
- On Solaris, finally fix the issue where queries get distributed strangely over CPUs, or not get distributed at all. Much debugging and analysing performed by Alex Kiernan, who also supplied fixes. Implemented in [commit 1091](#), [commit 1093](#).
- Various fixes for modern G++ versions, most spotted by Marcus Rueckert (commits [964](#), [965](#), [1028](#), [1052](#)), and Ruben Kerkhof ([commit 1136](#), closing [ticket 175](#)).
- Recursor would not properly clean up pidfile and control socket, closing [ticket 120](#), code in [commit 988](#), [commit 1098](#) (part of fix by Matti Hiljanen, spotted by Leo Baltus)

- Recursor can now serve multi-line records from its limited authoritative server ([commit 1014](#)).
- When parsing zones, the 'm' time specification stands for minutes, not months! Closing Debian bug 406462 ([commit 1026](#))
- Authoritative zone parser did not support '@' in the content of records. Spotted by Marco Davids, fixed in [commit 1030](#).
- Authoritative zone parser could be confused by trailing TABs on record lines ([commit 1062](#)).
- EINTR error code could block entire server if received at the wrong time. Spotted by Arnoud Bakker, fix in [commit 1059](#).
- Fix crash on NetBSD on Alpha CPUs, might improve startup behaviour on empty caches on other architectures as well ([commit 1061](#)).
- Outbound TCP queries were being performed sub-optimally because of an interaction with the 'MPlexer'. Fixes in [commit 1115](#), [commit 1116](#).

New features:

- Implemented **rec_control** command **get uptime**, as suggested by Niels Bakker ([commit 935](#)). Added to default rrdtool scripts in [commit 940](#).
- The Recursor Authoritative component, meant for having the Recursor serve some zones authoritatively, now supports \$INCLUDE and \$GENERATE. Implemented in [commit 951](#) and [commit 952](#), [commit 967](#) (discovered by Thomas Rietz),
- Implemented **forward-zones-file** option in order to support larger amounts of zones which should be forwarded to another nameserver ([commit 963](#)).
- Both **forward-zones** and **forward-zones-file** can now specify multiple forwarders per domain, implemented in [commit 1168](#), closing [ticket 81](#). Additionally, both these settings can also specify non-standard port numbers, as suggested in [ticket 122](#). Patch authored by Aaron Thompson, with additional work by Augie Schwer.
- Sten Spans contributed **allow-from-file**, implemented in [commit 1150](#). This feature allows the Recursor to read access rules from a (large) file.

General improvements:

- Ruben Kerkhof fixed up weird permission bits as well as our SGML documentation code in [commit 936](#) and [commit 937](#).
- Full IPv6 parity. If configured to use IPv6 for outgoing queries (using **query-local-address6:::0** for example), IPv6 and IPv4 addresses are finally treated 100% identically, instead of 'mostly'. This feature is implemented using 'ANY' queries to find A and AAAA addresses in one query, which is a new approach. Treat with caution.
- Now perform EDNS0 root refreshing queries, so as to benefit from all returned addresses. Relevant since early February 2008 when the root-servers started to respond with IPv6 addresses, which made the default non-EDNS0 maximum packet length reply no longer contain all records. Implemented in [commit 1130](#). Thanks to dns-operations AT mail.oarc.isc.org for quick suggestions on how to deal with this change.
- **rec_control** now has a timeout in case the Recursor does not respond. Implemented in [commit 945](#).
- (Error) messages are now logged with saner priorities ([commit 955](#)).
- Outbound query IP interface stemmed from 1997 (!) and was in dire need of a cleanup ([commit 1117](#)).
- L.ROOT-SERVERS.NET moved ([commit 1118](#)).

1.3.17 PowerDNS Authoritative Server version 2.9.21

Released the 21st of April 2007.

This is the first release the PowerDNS Authoritative Server since the Recursor was split off to a separate product, and also marks the transfer of the new technology developed specifically for the recursor, back to the authoritative server.

This move has reduced the amount of code of the Authoritative server by over 2000 lines, while improving the quality of the program enormously.

However, since so much has been changed, care should be taken when deploying 2.9.21.

To signify the magnitude of the underlying improvements, the next release of the PowerDNS Authoritative Server will be called 3.0.

This release would not have been possible without large amounts of help and support from the PowerDNS Community. We specifically want to thank Massimo Bandinelli of Italy's [Register.it](#), [Dave Aaldering of Aaldering ICT](#), [True BV](#), [XS4ALL](#), Daniel Bilik of [Neosystem](#), [EasyDNS](#), [Heinrich Ruthensteiner](#) of Siemens, [Augie Schwer](#), [Mark Bergsma](#), [Marco Davids](#), [Marcus Rueckert of OpenSUSE](#), Andre Muraro of [Locaweb](#), Antony Lesuisse, [Norbert Sendetzky](#), [Marco Chiavacci](#), Christoph Haas, Ralf van der Enden and Ruben Kerkhof.

Security issues:

- The previous packet parsing and generating code contained no known bugs, but was however very lengthy and overly complex, and might have had security problems. The new code is 'inherently safe' because it relies on bounds-checking C++ constructs. Therefore, a move to 2.9.21 is highly recommended.
- Pre-2.9.21, communication between master and server nameservers was not checked as rigidly as possible, possibly allowing third parties to disrupt but not modify such communications.



Warning

The 'bind1' legacy version of our BIND backend has been dropped! There should be no need to rely on this old version anymore, as the main BIND backend has been very well tested recently.

Bugs:

- Multi-part TXT records weren't supported. This has been fixed, and regression tests have been added. Code in commits [1016](#), [996](#), [994](#).
 - Email addresses with embedded dots in SOA records were not parsed correctly, nor were other embedded dots. Noted by 'Bastiaan', fixed in [commit 1026](#).
 - BIND backend treated the 'm' TTL modifier as 'months' and not 'minutes'. Closes Debian bug 406462. Addressed in [commit 1026](#).
 - Our snapshots were built against a static version of PostgreSQL that was incompatible with many Linux distributions, leading to instant crashes on startup. Fixed in [1022](#) and [1023](#).
 - CNAME referrals to child zones gave improper responses. Noted by Augie Schwer in [ticket 123](#), fixed in [commit 992](#).
 - When passing a port number with the **recursor** setting, this would sometimes generate errors during additional processing. Switched off overly helpful additional processing for recursive queries to remove this problem. Implemented in [commit 1031](#), spotted by Ralf van der Enden.
 - NS to a nameserver with the name of the zone itself generated problems. Spotted by Augie Schwer, fixed in [commit 947](#).
 - Multi-line records in the BIND backend were not always parsed correctly. Fixed in [commit 1014](#).
 - The LOC-record had problems operating outside of the eastern hemisphere of the northern part of the world! Fixed in [commit 1011](#).
-

- Backends were compiled without multithreading preprocessor flags. As far as we can determine, this would only cause problems for the BIND backend, but we cannot rule out this caused instability in other backends. Fixed in [commit 1001](#).
- The BIND backend was highly unstable under reloads, and leaked memory and file descriptors. Thanks to Mark Bergsma and Massimo Bandinelli for respectively pointing this out to us and testing large amounts of patches to fix the problem. The fixes have resulted in better performance, less code, and a remarkable simplification of this backend. Commits [1039](#), [1034](#), [1035](#), [1006](#), [999](#), [905](#) and previous.
- BIND backend gave convincing NXDOMAINs on unloaded zones in some cases. Spotted and fixed by Daniel Bilik in [commit 984](#).
- SOA records in zone transfers sometimes contained the wrong SOA TTL. Spotted by Christian Kuehn, fixed in [commit 902](#).
- PowerDNS could get confused by very high SOA serial numbers. Spotted and fixed by Dan Bilik, fixed in [commit 626](#).
- Some versions of FreeBSD perform very strict checks on socket address sizes passed to 'connect', which could lead to problems retrieving zones over AXFR. Fixed in [commit 891](#).
- Some versions of FreeBSD perform very strict checks on IPv6 socket addresses, leading to problems. Discovered by Sten Spans, fixed in [commit 885](#) and [commit 886](#).
- IXFR requests were not logged properly. Noted by Ralf van der Enden, fixed in [commit 990](#).
- Some NAPTR records needed an additional space character to encode correctly. Spotted by Heinrich Ruthensteiner, fixed in [commit 1029](#).
- Many bugs in the TCP nameserver, leading to a PowerDNS process that did not respond to TCP queries over time. Many fixes provided by Dan Bilik, other problems were fixed by rewriting our TCP handling code. Commits [982](#) and [980](#), [950](#), [924](#), [889](#), [874](#), [869](#), [685](#), [684](#).
- Fix crashes on the ARM processor due to alignment errors. Thanks to Sjoerd Simons. Closes Debian bug 397031.
- Missing data in generic SQL backends would sometimes lead to faked SOA serial data. Spotted by Leander Lakkas from True. Fix in [commit 866](#).
- When receiving two quick notifications in succession, the packet cache would sometimes "process" the second one, leading PowerDNS to ignore it. Spotted by Dan Bilik, fixed in [commit 686](#).
- Geobackend (by Mark Bergsma) did not properly override the getSOA method, breaking non-overlay operation of this fine backend. The geobackend now also skips '.hidden' configuration files, and now properly disregards empty configuration files. Additionally, the overlapping abilities were improved. Details available in [commit 876](#), by Mark.

Features:

- Thanks to [EasyDNS](#), PowerDNS now supports multiple masters per domain. For configuration details, see Section [18.2](#). Implemented in [commit 1018](#), [commit 1017](#).
- Thanks to [EasyDNS](#), PowerDNS now supports the KEY record type, as well the SPF record. In [commit 976](#).
- Added support for CERT, SSHFP, DNSKEY, DS, NSEC, RRSIG record types, as part of the move to the new DNS parsing/generating code.
- Support for the AFSDDB record type, as requested by 'Bastian'. Implemented in [commit 978](#), closing [ticket 129](#).
- Support for the MR record type. Implemented in [commit 941](#) and [commit 1019](#).
- Gsqlite3 backend was added by Antony Lesuisse in [commit 942](#);
- Added the ability to send out light-weight root-referrals that save bandwidth yet still placate mediocre resolver implementations. Implemented in [commit 912](#), enable with 'root-referral=lean'.

Improvements:

- Miscellaneous OpenDBX and LDAP backend improvements by Norbert Sendetzky. Applied in [commit 977](#) and [commit 1040](#).
- SGML source of the documentation was cleaned up by Ruben Kerkhof in [commit 936](#).
- Speedups in core DNS label processing code. Implemented in [commit 928](#), [commit 654](#), [commit 1020](#).
- When communicating with master servers and encountering errors, more useful details are logged. Reported by Stefan Arentz in [ticket 137](#), closed by [commit 1015](#).
- Database errors are now logged with more details. Addressed in [commit 1004](#).
- pdns_control problems are now logged more verbosely. Change in [commit 910](#).
- Erroneous address configuration was logged unclearly. Spotted by River Tarnell, fixed in [commit 888](#).
- Example configuration shipped with PowerDNS was very old. Noted by Leen Besselink, fixed in [commit 946](#).
- PowerDNS neglected to chdir to the root when chrooted. This closes [ticket 110](#), fixed in [commit 944](#).
- Microsoft resolver had problems with responses we generated for CNAMEs pointing out of our bailiwick. Fixed in [commit 983](#) and expedited by Locaweb.com.br.
- Built-in webserver logs errors more verbosely. Closes [ticket 82](#), fixed in [commit 991](#).
- Queries containing '@' no longer flood the logs. Addressed in [commit 1014](#).
- The build process now looks for PostgreSQL in more places. Implemented in [commit 998](#), closes [ticket 90](#).
- Speedups in the BIND backend now mean large installations enjoy startup times up to 30 times faster than with the original BIND nameserver. Many thanks to Massimo Bandinelli.
- BIND backend now offers full support for query logging, implemented in [commit 1026](#), [commit 1029](#).
- BIND backend named.conf parsing is now fully case-insensitive for domain names. This closes Debian bug 406461, fixed in [commit 1027](#).
- IPv6 and IPv4 address parsing routines have been replaced, which should result in prettier output in some cases. [commit 962](#), [commit 1012](#) and others.
- 5 new regression tests have been added to insure old bugs do not return.
- Fix small issues with very modern compilers and BOOST snapshots. Noted by Marcus Rueckert, addressed in [commit 954](#), [commit 964](#) [commit 965](#), [commit 1003](#).

1.3.18 Recursor version 3.1.4

Released the 13th of November 2006.

This release contains almost no new features, but consists mostly of minor and major bug fixes. It also addresses two major security issues, which makes this release a highly recommended upgrade.

Security issues:

- Large TCP questions followed by garbage could cause the recursor to crash. This critical security issue has been assigned CVE-2006-4251, and is fixed in [commit 915](#). More information can be found in [Section 1.5](#).
- CNAME loops with zero second TTLs could cause crashes in some conditions. These loops could be constructed by malicious parties, making this issue a potential denial of service attack. This security issue has been assigned CVE-2006-4252 and is fixed by [commit 919](#). More information can be found in [Section 1.6](#). Many thanks to David Gavarret for helping pin down this problem.

Bugs:

- On certain error conditions, PowerDNS would neglect to close a socket, which might therefore eventually run out. Spotted by Stefan Schmidt, fixed in commits [892](#), [897](#), [899](#).
- Some nameservers (including PowerDNS in rare circumstances) emit a SOA record in the authority section. The recursor mistakenly interpreted this as an authoritative "NXRRSET". Spotted by Bryan Seitz, fixed in [commit 893](#).
- In some circumstances, PowerDNS could end up with a useless (not working, or no longer working) set of nameserver records for a domain. This release contains logic to invalidate such broken NSSETs, without overloading authoritative servers. This problem had previously been spotted by Bryan Seitz, 'Cerb' and Darren Gamble. Invalidations of NSSETs can be plotted using the "nsset-invalidations" metric, available through `rec_control get`. Implemented in [commit 896](#) and [commit 901](#).
- PowerDNS could crash while dumping the cache using `rec_control dump-cache`. Reported by Wouter of WideXS and Stefan Schmidt and many others, fixed in [commit 900](#).
- Under rare circumstances (depleted TCP buffers), PowerDNS might send out incomplete questions to remote servers. Additionally, on big-endian systems (non-Intel and non-AMD generally), sending out large TCP answers questions would not work at all, and possibly crash. Brought to our attention by David Gavarret, fixed in [commit 903](#).
- The recursor contained the potential for a dead-lock processing an invalid domain name. It is not known how this might be triggered, but it has been observed by 'Cerb' on #powerdns. Several dead-locks where PowerDNS consumed all CPU, but did not answer questions, have been reported in the past few months. These might be fixed by [commit 904](#).
- IPv6 'allow-from' matching had problems with the least significant bits, sometimes allowing disallowed addresses, but mostly disallowing allowed addresses. Spotted by Wouter from WideXS, fixed in [commit 916](#).

Improvements:

- PowerDNS has support to drop answers from so called 'delegation only' zones. A statistic ("dlg-only-drops") is now available to plot how often this happens. Implemented in [commit 890](#).
- Hint-file parameter was mistakenly named "hints-file" in the documentation. Spotted by my Marco Davids, fixed in [commit 898](#).
- `rec_control quit` should be near instantaneous now, as it no longer meticulously cleans up memory before exiting. Problem spotted by Darren Gamble, fixed in [commit 914](#), closing [ticket 84](#).
- init.d script no longer refers to the Recursor as the Authoritative Server. Spotted by Wouter of WideXS, fixed in [commit 913](#).
- A potentially serious warning for users of the GNU C Library version 2.5 was fixed. Spotted by Marcus Rueckert, fixed in [commit 920](#).

1.3.19 Recursor version 3.1.3

Released the 12th of September 2006.

Compared to 3.1.2, this release again consists of a number of mostly minor bug fixes, and some slight improvements.

Many thanks are again due to Darren Gamble who together with his team has discovered many misconfigured domains that do work with some other name servers. DNS has long been tolerant of misconfigurations, PowerDNS intends to uphold that tradition. Almost all of the domains found by Darren now work as well in PowerDNS as in other name server implementations.

Thanks to some recent migrations, this release, or something very close to it, is powering over 40 million internet connections that we know of. We appreciate hearing about successful as well as unsuccessful migrations, please feel free to notify pdns.bd@powerdns.com of your experiences, good or bad.

Bug-fixes:

- The MThread default stack size was too small, which led to problems, mostly on 64-bit platforms. This stack size is now configurable using the `stack-size` setting should our estimate be off. Discovered by Darren Gamble, Sten Spans and a number of others. Fixed in [commit 868](#).
- Plug a small memory leak discovered by Kai and Darren Gamble, fixed in [commit 870](#).

- Switch from the excellent nedmalloc to dlmalloc, based on advice by the nedmalloc author. Nedmalloc is optimised for multithreaded operation, whereas the PowerDNS recursor is single threaded. The version of nedmalloc shipped contained a number of possible bugs, which are probably resolved by moving to dlmalloc. Some reported crashes on hitting 2G of allocated memory on 64 bit systems might be solved by this switch, which should also increase performance. See [commit 873](#) for details.

Improvements:

- The cache is now explicitly aware of the difference between authoritative and unauthoritative data, allowing it to deal with some domains that have different data in the parent zone than in the authoritative zone. Patch in [commit 867](#).
- No longer try to parse DNS updates as if they were queries. Discovered and fixed by Jan Gyselinck, fix in [commit 871](#).
- Rebalance logging priorities for less log cluttering and add IP address to a remote server error message. Noticed and fixed by Jan Gyselinck ([commit 877](#)).
- Add **logging-facility** setting, allowing syslog to send PowerDNS logging to a separate file. Added in [commit 871](#).

1.3.20 Recursor version 3.1.2

Released Monday 26th of June 2006.

Compared to 3.1.1, this release consists almost exclusively of bug-fixes and speedups. A quick update is recommended, as some of the bugs impact operators of authoritative zones on the internet. This version has been tested by some of the largest internet providers on the planet, and is expected to perform well for everybody.

Many thanks are due to Darren Gamble, Stefan Schmidt and Bryan Seitz who all provided excellent feedback based on their large-scale tests of the recursor.

Bug-fixes:

- Internal authoritative server did not differentiate between 'NXDOMAIN' and 'NXRRSET', in other words, it would answer 'no such host' when an AAAA query came in for a domain that did exist, but did not have an AAAA record. This only affects users with **auth-zones** configured. Discovered by Bryan Seitz, fixed in [commit 848](#).
- ANY queries for hosts where nothing was present in the cache would not work. This did not cause real problems as ANY queries are not reliable (by design) for anything other than debugging, but did slow down the nameserver and cause unnecessary load on remote nameservers. Fixed in [commit 854](#).
- When exceeding the configured maximum amount of TCP sessions, TCP support would break and the nameserver would waste CPU trying to accept TCP connections on UDP ports. Noted by Bryan Seitz, fixed in [commit 849](#).
- DNS queries come in two flavours: recursion desired and non-recursion desired. The latter is not very useful for a recursor, but is sometimes (erroneously) used by monitoring software or load balancers to detect nameserver availability. A non-rd query would not only not recurse, but also not query authoritative zones, which is confusing. Fixed in [commit 847](#).
- Non-standard DNS TCP queries, that did occur however, could drive the recursor to 100% CPU usage for extended periods of time. This did not disrupt service immediately, but does waste a lot of CPU, possibly exhausting resources. Discovered by Bryan Seitz, fixed in [commit 858](#), which is post-3.1.2-rc1.
- The PowerDNS recursor did not honour the rare but standardised 'ANY' query class (normally 'ANY' refers to the query type, not class), upsetting the Wildfire Jabber server. Discovered and debugged by Daniel Nauck, fixed in [commit 859](#), which is post-3.1.2-rc1.
- Everybody's favorite, when starting up under high load, a bogus line of statistics was sometimes logged. Fixed in [commit 851](#).
- Remove some spurious debugging output on dropping a packet by an unauthorized host. Discovered by Kai. Fixed in [commit 854](#).

Improvements:

- Misconfigured domains, with a broken nameserver in the parent zone, should now work better. Changes motivated and suggested by Darren Gamble. This makes PowerDNS more compliant with RFC 2181 by making it prefer authoritative data over non-authoritative data. Implemented in [commit 856](#).
- PowerDNS can now listen on multiple ports, using the **local-address** setting. Added in [commit 845](#).
- A number of speedups which should have a noticeable impact, implemented in commits [850](#), [852](#), [853](#), [855](#)
- The recursor now works around an issue with the Linux kernel 2.6.8, as shipped by Debian. Fixed by Christof Meerwald in [commit 860](#), which is post 3.1.2-rc1.

1.3.21 Recursor version 3.1.1



Warning

3.1.1 is identical to 3.1 except for a bug in the packet chaining code which would mainly manifest itself for IPv6 enabled Konqueror users with very fast connections to their PowerDNS installation. However, all 3.1 users are urged to upgrade to 3.1.1. Many thanks to Alessandro Bono for his quick aid in solving this problem.

Released on the 23rd of May 2006. Many thanks are due to the operators of some of the largest internet access providers in the world, each having many millions of customers, who have tested the various 3.1 pre-releases for suitability. They have uncovered and helped fix bugs that could impact us all, but are only (quickly) noticeable with such vast amounts of DNS traffic.

After version 3.0.1 has proved to hold up very well under tremendous loads, 3.1 adds important new features:

- Ability to serve authoritative data from 'BIND' style zone files (using **auth-zones** statement).
- Ability to forward domains so configured to external servers (using **forward-zones**).
- Possibility of 'serving' the contents of `/etc/hosts` over DNS, which is very well suited to simple domestic router/DNS setups. Enabled using **export-etc-hosts**.
- As recommended by recent standards documents, the PowerDNS recursor is now authoritative for RFC-1918 private IP space zones by default (suggested by Paul Vixie).
- Full outgoing IPv6 support (off by default) with IPv6 servers getting equal treatment with IPv4, nameserver addresses are chosen based on average response speed, irrespective of protocol.
- Initial Windows support, including running as a service ('NET START "POWERDNS RECURSOR"). **rec_channel** is still missing, the rest should work. Performance appears to be below that of the UNIX versions, this situation is expected to improve.

Bug fixes:

- No longer send out SRV and MX record priorities as zero on big-endian platforms (UltraSPARC). Discovered by Eric Sproul, fixed in [commit 773](#).
- SRV records need additional processing, especially in an Active Directory setting. Reported by Kenneth Marshall, fixed in [commit 774](#).
- The root-records were not being refreshed, which could lead to problems under inconceivable conditions. Fixed in [commit 780](#).
- Fix resolving domain names for nameservers with multiple IP addresses, with one of these addresses being lame. Other nameserver implementations were also unable to resolve these domains, so not a big bug. Fixed in [commit 780](#).
- For a period of 5 minutes after expiring a negative cache entry, the domain would not be re-cached negatively, leading to a lot of duplicate outgoing queries for this short period. This fix has raised the average cache hit rate of the recursor by a few percent. Fixed in [commit 783](#).

- Query throttling was not aggressive enough and not all sorts of queries were throttled. Implemented in [commit 786](#).
- Fix possible crash during startup when parsing empty configuration lines ([commit 807](#)).
- Fix possible crash when the first query after wiping a cache entry was for the just deleted entry. Rare in production servers. Fixed in [commit 820](#).
- Recursor would send out differing TTLs when receiving a misconfigured, standards violating, RRSET with different TTLs. Implement fix as mandated by RFC 2181, paragraph 5.2. Reported by Stephen Harker ([commit 819](#)).
- The **top-remotes** would list remotes more than once, once per source port. Discovered by Jorn Ekkelenkamp, fixed in [commit 827](#), which is post 3.1-pre1.
- Default **allow-from** allowed queries from fe80::/16, corrected to fe80::/10. Spotted by Niels Bakker, fixed in [commit 829](#), which is post 3.1-pre1.
- While PowerDNS blocks failing queries quickly, multiple packets could briefly be in flight for the same domain and nameserver. This situation is now explicitly detected and queries are chained to identical queries already in flight. Fixed in [commit 833](#) and [commit 834](#), post 3.1-pre1.

Improvements:

- ANY queries are now implemented as in other nameserver implementations, leading to a decrease in outgoing queries. The RFCs are not very clear on desired behaviour, what is implemented now saves bandwidth and CPU and brings us in line with existing practice. Previously ANY queries were not cached by the PowerDNS recursor. Implemented in [commit 784](#).
- **rec_control** was very sparse in its error reporting, and user unfriendly as well. Reported by Erik Bos, fixed in [commit 818](#) and [commit 820](#).
- IPv6 addresses were printed in a non-standard way, fixed in [commit 788](#).
- TTLs of records are now capped at two weeks, [commit 820](#).
- **allow-from** IPv4 netmasks now automatically work for IP4-to-IPv6 mapped IPv4 addresses, which appear when running on the wildcard :: IPv6 address. Lack of feature noted by Marcus 'darix' Rueckert. Fixed in [commit 826](#), which is post 3.1-pre1.
- Errors before daemonizing are now also sent to syslog. Suggested by Marcus 'darix' Rueckert. Fixed in [commit 825](#), which is post 3.1-pre1.
- When launching without any form of configured network connectivity, all root-servers would be cached as 'down' for some time. Detect this special case and treat it as a resource-constraint, which is not accounted against specific nameservers. Spotted by Seth Arnold, fixed in [commit 835](#), which is post 3.1-pre1.
- The recursor now does not allow authoritative servers to keep supplying its own NS records into perpetuity, which causes problems when a domain is redelegated but the old authoritative servers are not updated to this effect. Noticed and explained at length by Darren Gamble of Shaw Communications, addressed by [commit 837](#), which is post 3.1-pre2.
- Some operators may want to follow RFC 2181 paragraph 5.2 and 5.4. This harms performance and does not solve any real problem, but does make PowerDNS more compliant. If you want this, enable **auth-can-lower-ttl**. Implemented in [commit 838](#), which is post 3.1-pre2.

1.3.22 Recursor version 3.0.1

Released 25th of April 2006, [download](#).

This release consists of nothing but tiny fixes to 3.0, including one with security implications. An upgrade is highly recommended.

- Compilation used both `cc` and `gcc`, leading to the possibility of compiling with different compiler versions ([commit 766](#)).
- **rec_control** would leave files named `lsockXXXXXX` around in the configured socket-dir. Operators may wish to remove these files from their socket-dir (often `/var/run`), quite a few might have accumulated already ([commit 767](#)).

- Certain malformed packets could crash the recursor. As far as we can determine these packets could only lead to a crash, but as always, there are no guarantees. A quick upgrade is highly recommended (commits [760](#), [761](#)). Reported by David Gavarret.
- Recursor would not distinguish between NXDOMAIN and NXRRSET ([commit 756](#)). Reported and debugged by Jorn Ekkelenkamp.
- Some error messages and trace logging statements were improved (commits [756](#), [758](#), [759](#)).
- stderr was closed during daemonizing, but not dupped to /dev/null, leading to slight chance of odd behaviour on reporting errors ([commit 757](#)).

Operating system specific fixes:

- The stock Debian sarge Linux kernel, 2.6.8, claims to support epoll but fails at runtime. The epoll self-testing code has been improved, and PowerDNS will fall back to a select based multiplexer if needed ([commit 758](#)) Reported by Michiel van Es.
- Solaris 8 compilation and runtime issues were addressed. See the README for details ([commit 765](#)). Reported by Juergen Georgi and Kenneth Marshall.
- Solaris 10 x86_64 compilation issues were addressed ([commit 755](#)). Reported and debugged by Eric Sproul.

1.3.23 Recursor version 3.0

Released 20th of April 2006, [download](#).

This is the first separate release of the PowerDNS Recursor. There are many reasons for this, one of the most important ones is that previously we could only do a release when both the recursor and the authoritative nameserver were fully tested and in good shape. The split allows us to release new versions when each part is ready.

Now for the real news. This version of the PowerDNS recursor powers the network access of over two million internet connections. Two large access providers have been running pre-releases of 3.0 for the past few weeks and results are good. Furthermore, the various pre-releases have been tested nearly non-stop with DNS traffic replayed at 3000 queries/second.

As expected, the 2 million households shook out some very rare bugs. But even a rare bug happens once in a while when there are this many users.

We consider this version of the PowerDNS recursor to be the most advanced resolver publicly available. Given current levels of spam, phishing and other forms of internet crime we think no recursor should offer less than the best in spoofing protection. We urge all operators of resolvers without proper spoofing countermeasures to consider PowerDNS, as it is a Better Internet Nameserver Daemon.

A good article on DNS spoofing can be found [here](#). Some more information, based on a previous version of PowerDNS, can be found on the [PowerDNS development blog](#).



Warning

Because of recent DNS based denial of service attacks, running an open recursor has become a security risk. Therefore, unless configured otherwise this version of PowerDNS will only listen on localhost, which means it does not resolve for hosts on your network. To fix, configure the **local-address** setting with all addresses you want to listen on. Additionally, by default service is restricted to RFC 1918 private IP addresses. Use **allow-from** to selectively open up the recursor for your own network. See Section [17.1](#) for details.

Important new features of the PowerDNS recursor 3.0:

- Best spoofing protection and detection we know of. Not only is spoofing made harder by using a new network address for each query, PowerDNS detects when an attempt is made to spoof it, and temporarily ignores the data. For details, see Section [17.5.1](#).
 - First nameserver to benefit from epoll/kqueue/Solaris completion ports event reporting framework, for stellar performance.
 - Best statistics of any recursing nameserver we know of, see Section [17.6](#).
-

- Last-recently-used based cache cleanup algorithm, keeping the 'best' records in memory
- First class Solaris support, built on a 'try and buy' Sun CoolThreads T 2000.
- Full IPv6 support, implemented natively.
- Access filtering, both for IPv4 and IPv6.
- Experimental SMP support for nearly double performance. See Section [17.4](#).

Many people helped package and test this release. Jorn Ekkelenkamp of ISP-Services helped find the '8000 SOAs' bug and spotted many other oddities and [XS4ALL](#) internet funded a lot of the recent development. Joaquín M López Muñoz of the `boost::multi_index_container` was again of great help.

1.3.24 Version 2.9.20

Released the 15th of March 2006

Besides adding OpenDBX, this release is mostly about fixing problems and speeding up the recursor. This release has been made possible by [XS4ALL](#) and [True](#). Thanks!

Furthermore, we are very grateful for the help of Andrew Pinski, who hacks on gcc, and of Joaquín M López Muñoz, the author of `boost::multi_index_container`. Without their near-realtime help this release would've been delayed a lot. Thanks!

Bugs fixed in the recursor:

- Possible stability issues in the recursor on encountering errors ([commit 532](#), [commit 533](#))
- Memory leaks in recursor fixed ([commit 534](#), [commit 572](#)). In a test 800 million real life DNS packets have been sent to the recursor, representing several days of traffic from a major ISP, memory use was high (500MB), but stable.
- Prune all data in PowerDNS - previously per-nameserver and per-query performance statistics were kept around forever ([commit 535](#))
- IPv6 additional processing was broken. Reported by Lionel Elie Mamane, who also provided a fix. The problem was fixed differently in the end. [commit 562](#).
- `pdns_recursor` did not shuffle answers since 2.9.19, leading to problems sending mail to the Hotmail servers. Reported in [ticket 54](#), fixed in [commit 567](#).
- If a single nameserver had multiple IP addresses listed, PowerDNS would only use one of them. Noted by Mark Martin, fixed in [commit 570](#), who depends on a domain with 4 nameserver IP addresses of which 2 are broken.

Improvements to the recursor:

- Commits [535](#), [540](#), [541](#), [542](#), [543](#), [544](#), [545](#), [547](#) and [548](#), [574](#) all speed up the recursor by a large factor, without altering the DNS algorithm.
- Move recursor to the incredible `boost::multi_index_container` ([commit 580](#)). This brings a huge improvement in cache pruning times.
- [commit 549](#) and [commit 550](#) work around gcc bug [24704](#) if requested, which speeds up the recursor a lot, but involves a dirty hack. Enable with `./configure --enable-gcc-skip-locking`. No guarantees!

Bugs fixed in the authoritative nameserver:

- PowerDNS would no longer allow a '/' in domain names, fixed by [commit 537](#), reported in [ticket 48](#).
- Parameters to `pdns_control notify-host` were not checked, leading to possible crashes. Reported in [ticket 24](#), fixed in [commit 565](#).

- On some compilers, processing of NAPTR records could cause the server to crash. Reported by Bernd Froemel in [ticket 29](#), fixed in [commit 538](#).
- Backend errors could make the whole nameserver exit under some circumstances, notably using the LDAP backend. Fixed in [commit 583](#), reported in [ticket 62](#).
- Referrals were subtly broken by recent CNAME/Wildcard improvements, fixed in [commit 539](#). Fix and other improvements sponsored by [True](#).
- PowerDNS would try to insert records it has no knowledge about in slave zones, which did not work. Reported in [ticket 60](#), fixed in [commit 566](#). A superior fix would be to implement the relevant unknown record standard.

Improvements to the authoritative nameserver:

- Pipebackend did not properly propagate the ABI version to its children, fixed in [commit 546](#), reported by kickdaddy@gmail.com in [ticket 45](#).
- [OpenDBX](#) backend added ([commit 559](#), [commit 560](#), [commit 561](#)) by Norbert Sendetzky. From the website: ‘ The OpenDBX backend enables it to fetch DNS information from every DBMS supported by the OpenDBX library and combines the power of one of the best DNS server implementations with the flexibility of the OpenDBX library. ’ OpenDBX adds some other features like database failover. Thanks Norbert!
- LDAP fixes as reported in [ticket 37](#), fixed in [commit 558](#), which make `pdns_control notify` work.
- Arjo Hooimeijer added support for `soa-refresh-default`, `soa-retry-default`, `soa-expire-default`, which were previously hardcoded. [commit 563](#) and fallout in [commit 573](#) (thanks to Wolfram Schlich).

Miscellaneous:

- Fixes for g++ 4.1. Compiling with 4.1 realizes notable speedups. [commit 568](#), [commit 569](#).
- PowerDNS now reports if it is running in 32 or 64 bit mode, useful for bi-arch users that need to know if they are benefitting from [AMD's great processor](#). [commit 571](#).
- `dnsscope` compiles again, [commit 551](#), [commit 564](#) (FreeBSD 64-bit `time_t`).
- `dnsreplay_mindex` compiles again, fixed by [commit 572](#). Its performance, and the performance of the recursor was improved by [commit 559](#).
- Build scripts were added, mostly for internal use but we know some PowerDNS users build their own packages too. [commit 553](#), [commit 554](#), [commit 555](#), [commit 556](#), [commit 557](#).
- `bootstrap` script was not included in release. Thanks to Stefan Arentz for noticing. Fixed in [commit 574](#).

1.3.25 Version 2.9.19

Released 29th of October 2005.

As with other recent releases, the usage of PowerDNS appears to have skyrocketed. Informal, though strict, measurements show that PowerDNS now powers around 50% of all German domains, and somewhere in the order of 10-15% of the rest of the world. Furthermore, DNS is set to take a central role in connecting Voice over IP providers, with PowerDNS offering a very good feature set for these ENUM deployments. PowerDNS is already powering the E164.info ENUM zone and also acts as the backend for a major VoIP provisioning platform.

Included in this release is the now complete packet parsing/generating, record parsing/generating infrastructure. Furthermore, this framework is used by the recursor, hopefully making it very fast, memory efficient and robust. Many records are now processed using a single line of code. This has made the recursor a lot stricter in packet parsing, you will see some error messages which did not appear before. Rest assured however that these only happen for queries which have no valid answer in any case.

Furthermore, support for DNSSEC records is available in the new infrastructure, although it should be emphasised that there is more to DNSSEC than parsing records. There is no real support for DNSSEC (yet).

Additionally, the BIND Backend has been replaced by what was up to now known as the 'Bind2Backend'. Initial benchmarking appears to show that this backend is faster, uses less memory and has shorter startup times. The code is also shorter.

This release fixes a number of embarrassing bugs and is a recommended upgrade.

Thanks are due to [XS4ALL](#) who are supporting continuing development of PowerDNS, the fruits of which can be found in this release already. Furthermore, a remarkable number of people have helped report bugs, validate solutions or have submitted entire patches. Many thanks!

Improvements:

- `dnsreplay` now has a help message and has received further massive updates, making the code substantially faster. It turns out that `dnsreplay` is often 'heavier' than the PowerDNS process being benchmarked.
- PowerDNS recursor no longer prints out its queries by default as most recursor deployments have too much traffic for this to be useful.
- PowerDNS recursor is now able to read its root-hints from disk, which is useful to operate with alternate roots, like the [Open Root Server Network](#). See [Chapter 17](#).
- PowerDNS can now send out old-fashioned root-referrals when queried for domains for which it is not authoritative. Wastes some bandwidth but may solve incoming query floods if domains are delegated to you for which you are not authoritative, but which are queried by broken recursors.
- PowerDNS now prints out a warning when running with legacy LinuxThreads implementation instead of the high performance NPTL library, see [Section 9.2](#). [commit 455](#).
- A lot of superfluous calls to `gettimeofday()` have been removed, making PowerDNS and especially the recursor faster. Suggested by Kai.
- SPF records are now supported natively. [commit 472](#), closing [ticket 22](#).
- Improved IPv6 'bound to' messages. Thanks to Niels Bakker, Wichert Akkerman and Gerty de Wolf for suggestions.
- Separate graphs can now be made of IPv6 queries and answers. [commit 485](#).
- Out of zone additional processing is now on by default to better comply with standards. [commit 487](#).
- Regression tests have been expanded to deal with more record types (SRV, NAPTR, TXT, duplicate SRV).
- Improved query-logging in Bindbackend, which can be used for debugging purposes.
- Dropped `libpcap` dependency, making compilation easier
- `pdns_control` now has a help message.
- Add RRSIG, DNSKEY, DS and NSEC records for DNSSEC-bis to new parser infrastructure.
- Recursor now honours EDNS0 allowing it to send out larger answers.

Bugs fixed:

- Domain name validation has been made a lot stricter - it turns out PostgreSQL was interpreting some (corrupt) domain names as unicode. Tested and suggested by Register.com ([commit 451](#)).
- LDAP backend did not compile (commits [452](#), [453](#)) due to partially applied patch (Norbert Sendetzky)
- Incoming zone transfers work reliably again. Fixed in [commit 460](#) and beyond. And [commit 523](#) - closing Debian bug 330184.
- Recent g++ versions exposed a mistake in the PowerDNS recursor cache pruning code, causing random crashes. Fixed in [commit 465](#). Reported by several Red Hat users.
- PowerDNS recursor, and MTasker in general, did not work on Solaris. Patch by Juergen Ilse, [commit 471](#). Also moved most of PowerDNS over to `uint32_t` style typedefs, which eases compilation problems on Solaris, [commit 477](#).

- Bindbackend2 did not properly search its include path for \$INCLUDE statements. Noted by Mark Bergsma, [commit 474](#).
- Bindbackend did not notice changed zones, this problem has been fixed by the move to Bind2.
- Pipebackend did not clean up, leading to an additional pipe backend per AXFR or pdns_control reload. Discovered by Marc Jauvin, fixed by [commit 525](#).
- Bindbackend (both old and current versions) did not honour 'include' statements in `named.conf` on **pdns_control rediscover**. Noted by Marc Jauvin, fixed by [commit 526](#).
- Zone transfers were sometimes shuffled, which wastes useless time, [commit 478](#).
- CNAMEs and Wildcards now work as in Bind, fixing many complaints, [commit 487](#).
- NAPTR records were compressed, which would work, but was in violation of the RFC, [commit 493](#).
- NAPTR records were not always parsed correctly from BIND zone files, fixed, [commit 494](#).
- Geobackend needed additional include statement to compile on more recent Linux distributions, [commit 496](#).

1.3.26 Version 2.9.18

Released on the 16th of July 2005.

The '8 million domains' release, which also marks the battle readiness of the PowerDNS Recursor. The latest improvements have been made possible by financial support and contributions by [Register.com](#) and [XS4ALL](#). Thanks!

This release brings a number of new features (vastly improved recursor, Generic Oracle Support, DNS analysis and replay tools, and more) but also has a new build dependency, the [Boost library](#) (version 1.31 or higher).

Currently several big ISPs are evaluating the PowerDNS recursor for their resolving needs, some of them have switched already. In the course of testing, over 350 million actual queries have been recorded and replayed, the answers turn out to be satisfactorily.

This testing has verified that the pdns recursor, as shipped in this release, can stand up to heavy duty ISP loads (over 20000 queries/second) and in fact does so better than major other nameservers, giving more complete answers and being faster to boot.

We invite ISPs who note recursor problems to record their problematic traffic and replay it using the tools described in [Chapter 25](#) to discover if PowerDNS does a better job, and to let us know the results.

Additionally, the bind2backend is almost ready to replace the stock bind backend. If you run with Bind zones, you are cordially invited to substitute 'launch=bind2' for 'launch=bind'. This will happen automatically in 2.9.19!

In other news, the entire Wikipedia constellation now runs on PowerDNS using the Geo Backend! Thanks to Mark Bergsma for keeping us updated.

There are two bugs with security implications, which only apply to installations running with the LDAP backend, or installations providing recursion to a limited range of IP addresses. If any of these apply to you, an upgrade is highly advised:

- The LDAP backend did not properly escape all queries, allowing it to fail and not answer questions. We have not investigated further risks involved, but we advise LDAP users to update as quickly as possible (Norbert Sendetzky, Jan de Groot)
- Questions from clients denied recursion could blank out answers to clients who are allowed recursion services, temporarily. Reported by Wilco Baan. This would've made it possible for outsiders to blank out a domain temporarily to your users. Luckily PowerDNS would send out SERVFAIL or Refused, and not a denial of a domain's existence.

General bugs fixed:

- TCP authoritative server would not relaunch a backend after failure (reported by Norbert Sendetzky)
 - Fix backend restarting logic (reported, and fix suggested by Norbert Sendetzky)
 - Launching identical backends multiple times, with different settings, did not work. Reported by Mario Manno.
 - Master/slave queries did not honour the **query-local-address** setting. Spotted by David Levy of Register.com. The fix also randomises the local port used, slightly improving security.
-

Compilation fixes:

- Fix compile on Solaris, they define 'PC' for some reason. Reported by Eric Yiu.
- PowerDNS recursor would not compile on FreeBSD due to Linux specific defines, as reported in cvstrac ticket 26 (Ralf van der Enden)
- Several 64 bits issues have been fixed, especially in the Logging subsystem.
- SQLite would fail to compile on recent Debian systems (Matthijs Möhlmann)
- Generic MySQL would not compile on 64-bit platforms.

Improvements:

- PowerDNS now reports stray command line arguments, like when running '--local-port 5300' instead of '--local-port=5300'. Reported by Christian Welzel.
- We now warn against erroneous logging-facility specification, ie specifying an unknown facility.
- **--version** now outputs gcc version used, so we can tell people 2.95 is no longer supported.
- Extended regression tests, moved them to the new 'sdig' tool (see below).
- Bind2backend is now blazingly fast, and highly memory efficient to boot. As a special bonus it can read gzipped zones directly. The '.NET' zone is hosted using 401MB of memory, the same size as the zone on disk.
- The Pipe Backend has been improved such that it can send out different answers based on the IP address the question was received ON. See Section [A.1.1](#) for how this changed the Pipe Backend protocol. Note that you need to set **pipebackend-abi-version** to benefit from this change, existing clients are not affected. Change and documentation contributed by Marc Jauvin of Register4Less.
- LDAP backend has been updated (Norbert Sendetzky).

Recursor improvements and fixes. See Chapter [16](#) for details. The changes below mean that all of the caveats listed for the recursor have now been addressed.

- After half an hour of uptime, the entire cache would be pruned for each packet, which is a tad slow. It now appears the pdns recursor is among the fastest around.
 - Under high loads, or when unlucky, some query mthreads would get 'stuck', and show up in the statistics as eternally running queries.
 - Lots of redundant gettimeofday() and time() calls were removed, which has resulted in a measurable speedup.
 - pdns_recursor can now listen on several addresses simultaneously.
 - Now supports setuid and setgid operation to allow running as a less privileged user (Bram Vandoren).
 - Return code of pdns_recursor binary did not make sense (Matthijs Möhlmann and Thomas Hood)
 - Timeouts and errors are now split out in statistics.
 - Many people reported broken statistics, it turned out that no statistics were being reported if there had been no questions to base them on. We now log a message to that effect.
 - Add **query-local-address** support, which allows the recursor to send questions from a specific IP address. Useful for anycast setups.
 - Add outgoing TCP query support and proper truncated answer support. Needed for Worldnic Denial of Service protection, which sends out truncated packets to force clients to connect over TCP, which prevents spoofing.
 - Properly truncate our own answers.
-

- Improve our TCP answers by using `writenv`, which is slightly friendlier to the network.
- On FreeBSD, TCP errors could cause the recursor to exit suddenly due to a SIGPIPE signal.
- Maximum number of simultaneous client TCP connections can now be limited with the **`max-tcp-clients`** setting.
- Add aggressive timeouts for TCP clients to make sure resources are not wasted. Defaults to two seconds, can be configured with the **`client-tcp-timeout`** setting.

Backend fixes:

- SQLite backend would not slave properly (Darron Broad)
- Generic MySQL would not compile on 64-bit platforms.

New technology:

- Added the new DNS parser logic, called MOADNSParser. Completely modular, every memory access checked.
- `'sdig'`, a simple dig work-alike with `'canonical'` output, which is used for the regression tests. Based on the new DNS parser logic.
- **`dnswasher`**, **`dnsreplay`** and **`dnsscope`**, all DNS analysis tools. See Chapter 25 for more details.
- Generic Oracle Backend, sponsored by Register.COM. See Section A.3.3.

1.3.27 Version 2.9.17

See [the new timeline](#) for progress reports.

The 'million domains' release - PowerDNS has now firmly established itself as a major player with the unofficial count (ie, guesswork) now at over two million PowerDNS domains! Also, the GeoBackend has been tested by a big website and may soon see wider deployment. Thanks to Mark Bergsma for spreading the word!

It is also a release with lots of changes and fixes. Take care when deploying!

Security issues:

- PowerDNS could be temporarily DoSed using a random stream of bytes. Reported cause of this has been fixed.

Enhancements:

- Reported version can be changed, or removed - see the `"version-string"` setting.
- Duplicate MX records are now no longer considered duplicate if their priorities differ. Some people need this feature for spam filtering.

Bug fixes:

- NAPTR records can now be slaved, patch by Lorens Kockum.
 - GMySQL now works on Solaris
 - PowerDNS could be confused by questions with a `%`-sign in them - fixing cvstrac ticket #16 (reported by dilinger at voxel.net)
 - An authentication bug in the webserver was possibly fixed, please report if you were suffering from this. Being unable to authenticate to the webserver was what you would've noticed.
 - Fix for cvstrac ticket #2, PowerDNS could lose sync when sending out a very large number of notifications. Excellent bug report by Martin Hoffman, who also improved our original bugfix.
-

- Fix the oldest PowerDNS bug in existence - under some circumstances, PowerDNS would log to syslog one character at a time. This was cvstrac ticket #4
- HINFO records can now be slaved, fixing cvstrac ticket #8.
- pdns_recursor could block under some circumstances, especially in case of corrupt UDP packets. Reported by Wichert Akkerman. Fix by Christopher Meer. This was cvstrac ticket #13.
- Large SOA serial numbers would sometimes be logged as a signed integer, leading to negative numbers in the log.
- PowerDNS now fully supports 32 bit SOA serial numbers (thanks to Mark Bergsma), closing cvstrac ticket #5.
- pdns_recursor --local-address help text was wrong.
- Very devious bug - PowerDNS did not clear its cache before sending out update notifications, leading slaves to conclude there was no update to AXFR. Excellent debugging by mkuchar at wproduction.cz.
- Probably fixed cvstrac ticket #26, which caused pdns_recursor to fail on recent FreeBSD 5.3 systems. Please check, I have no such system to test on.
- Geobackend did not get built for Debian.

1.3.28 Version 2.9.16

The 'it must still be Friday somewhere' release. Massive number of fixes, portability improvements and the new Geobackend by Mark Bergsma & friends.

New:

- The Geobackend which makes it possible to send different answers to different IP ranges. Initial documentation can be found in pdns/modules/geobackend/README.
- qgen query generation tool. Nearly completely undocumented and hard to build too, it requires Boost. But very spiffy. Use **cd pdns; make qgen** to build it.

Bugfixes:

- The most reported bug ever was fixed. Zone2sql required the inclusion of unistd.h, except on Debian unstable.
- PowerDNS tried to listen on its control "pipe" which does not work. Probably harmless, but might have caused some oddities.
- The Packet Cache did not always set its TTL immediately, causing some packets to be inserted, even when running with the cache disabled (Mark Bergsma).
- Valgrind found some uninitialized reads, causing bogus values in the priority field when it was not needed.
- Valgrind found a bug in MTasker where we used delete instead of delete[].
- SOA serials and other parameters are unsigned. This means that very large SOA serial numbers would be messed up (Michel Stol, Stefano Straus)
- PowerDNS left its controlsocket around after exit and reported confusing errors if a socket was already in use.
- The recursor proxy did not work on big endian systems like SPARC and some MIPS processors (Remco Post)
- We no longer dump core on processing LOC records on UltraSPARC (Andrew Mulholland supplied a testing machine)

Improvements:

- MySQL can now connect to a specified port again (Chris Anderton).
-

- When running chroot(ed) and with master or slave support active, PowerDNS needs to resolve domain names to find slaves. This in turn may require access to certain libraries. Previously, these needed to be available in the chroot directory but by forcing an initial lookup, these libraries are now loaded before the chrooting.
- pdns_recursor was very slow after having done a larger number of queries because of the checks to see if a query should be throttled. This is now done using a set which is a lot faster than the previous full sequential scan.
- The throttling code may not have throttled as much as was configured.
- Yet another big LDAP update. The LDAP backend now load balances connections over several hosts (Norbert Sendetzky)
- Updated b.root-servers.net address in the recursor

1.3.29 Version 2.9.15

This release fixes up some of the shortcomings in 2.9.14, and adds some new features too.

Bugfixes:

- **allow-recursion-override** was on by default, it was meant to be off.
- Logging was still off in daemon mode, fixed.
- debian/rules forgot to build an sqlite package
- Recursor accidentally linked in MySQL - this was the result of an experiment with a persistent recursor cache.
- The PowerDNS recursor had stability problems. It now sorts nameservers (roughly) by responsiveness. The 'roughly' part upset the sorting algorithm used, the speeds being sorted on changed during sorting.
- The recursor now outputs the nameserver average response times in trace mode
- LDAP compiles again.

Improvements:

- zone2sql can now accept - as a file name which causes it to read stdin. This allows the following to work: **dig axfr ds9a.nl | zone2sql --gmysql --zone=- | mysql pdns**, which is a nice way to import a zone.
- zone2sql now ignores duplicate SOA records which are identical - which also makes the above possible.
- Remove libpqpp dependencies - since we now use the native C API for PostgreSQL

1.3.30 Version 2.9.14

Big release with the fix for the all important 2^30 seconds problem and a lot of other news.

- errno problems would cause compilation problems when using LDAP (Norbert Sendetzky)
 - The Generic SQL backend could cause crashes on PostgreSQL when using pdns_control notify (Georg Bauer)
 - Debian compatible init.d script (Wichert Akkerman)
 - If using the master or slave features, pdns had the notion of eternity ending in 2038, except that due to a thinko, eternity ended out to be the 10th of January 2004. This caused a loop to timeout immediately. Many thanks to Jasper Spaans for spotting the bug within five minutes.
 - Parts of the SOA field were not canonicalized.
 - The loglevel could in fact cause nothing to be logged (Norbert Sendetzky)
-

Improvements:

- The recursor now chooses the fastest nameserver, which causes a big speedup!
- LDAP now has different lookup models
- Cleanups, better load distribution, better exception handling, zone2ldap improvements
- The recursor was somewhat chatty about TCP connections
- PostgreSQL now only depends on the C API and not on the deprecated C++ one
- PowerDNS can now fully overrule external zones when doing recursion. See Chapter 16.

1.3.31 Version 2.9.13

Big news! Windows is back! Our great friend Michel Stol found the time to update the PowerDNS code so it works again under windows.

Furthermore, big thanks go out to Dell who quickly repaired my trusty **laptop**.

His changes:

- Generic SQLite support added
- Removed the ODBC backend, replaced it by the Generic ODBC Backend, which has all the cool configurability of the Generic MySQL and PostgreSQL backends.
- The PowerDNS Recursor now runs as a Service. It defaults to running on port 5300, PowerDNS itself is configured to expect the Recursor on port 5300 now.
- The PowerDNS Service is now known as 'PowerDNS' to Windows.
- The Installer was redone, this time with **NSIS2**.
- General updates and fixes.

Other news:

Note

There appears to be a problem with PowerDNS on Red Hat 7.3 with GCC 2.96 and self-compiled binaries. The symptoms are that PowerDNS works on the foreground but fails as a daemon. We're working on it.

If you do note problems, let the list know, if you don't, please do so as well. Tell us if you use the RPM or compiled yourself. It is known that not compiling in MySQL support helps solve the problem, but then you don't have MySQL.

There have been a number of reports on MySQL connections being dropped on FreeBSD 4.x, which sometimes causes PowerDNS to give up and reload itself. To combat this, MySQL error messages have been improved in some places in hopes of figuring out what is up. The initial indication is that MySQL itself sometimes terminates the connection and, amazingly, that switching to a Unix domain socket instead of TCP solves the problem.

Bug fixes:

- **allow-axfr-ips** did not work for individual IP addresses (bug & fix by Norbert Sendetzky)

Improvements:

- Opteron support! Thanks to Jeff Davey for providing a shell on an Opteron. The fixes should also help PowerDNS on other platforms with a 64 bit userspace.

Btw, the PowerDNS team has a strong desire for an Opteron :-)

- `pdns_recursor` jumbles answers now. This means that you can do poor man's round robin by supplying multiple A, MX or AAAA records for a service, and get a random one on top each time. Interestingly, this feature appeared out of nowhere, this change was made to the authoritative code but due to the wonders of code-reuse had an effect on `pdns_recursor` too.
- Big LDAP cleanup. Support for TLS was added. `Zone2LDAP` also gained the ability to generate `ldif` files containing a tree or a list of entries. (Norbert Sendetzky)
- `Zone2sql` is now somewhat clearer when reporting malformed line errors - it did not always include the name of the file causing a problem, especially for big installations. Problem noted by Thom May.
- `pdns_recursor` now survives the expiration of all its root records, most often caused by prolonged disconnection from the net.

1.3.32 Version 2.9.12

Release rich in features. Work on Verisign oddities, addition of SQLite backend, `pdns_recursor` maturity.

New features:

- `--version` command (requested by Mike Benoit)
- `delegation-only`, a Verisign special.
- Generic [SQLite](#) support, by Michel 'Who da man?' Stol. See [Section A.5](#).
- `init.d` script for `pdns_recursor`
- Recursor now actually purges its cache, saving memory.
- Slave configuration now no longer falls over when presented with a NULL master
- `Bindbackend2` now has supermaster support (Mark Bergsma, untested)
- Answers are now shuffled! It turns out a few recursors don't do shuffling (`pdns_recursor`, `djbdns`), so we do it now. Requested by Jorn Ekkelenkamp of ISP-Services. This means that if you have multiple IP addresses for one host, they will be returned in differing order every once in a while.

Bugs:

- 0.0.0.0/0 didn't use to work (Norbert Sendetzky)
- `pdns_recursor` would try to resolve IP address which to bind to, potentially causing chicken/egg problem
- `gpgsql` no longer reports as `gmysql` (Sherwin Daganoto)
- SRV would not be parsed right from disk (Christof Meerwald)
- An AXFR from a zone hosted on the LDAP backend no longer transmits all the reverse entries too (Norbert Sendetzky)
- PostgreSQL backend now does error checking. It would be a bit too trusting before.

Improvements, cleanups:

- PowerDNS now reports the numerical IP addresses it binds to instead of the, possibly, alphanumeric names the operator passed.
 - Removed `only-soa` hackery (noticed by Norbert Sendetzky)
 - Debian packaging fixes (Wichert Akkerman)
 - Some parameter descriptions were improved.
 - Cleanups by Norbert: `getAuth` moved to `chopOff`, `arguments::contains` massive cleanup, more.
-

1.3.33 Version 2.9.11

Yet another iteration, hopefully this will be the last silly 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.

This release enables AXFR again, **disable-axfr** did the opposite of what it claimed. Furthermore, the `pdns_recursor` now cleans its cache, which should save some memory in the long run. Norbert contributed some small LDAP work which should come in useful in the future.

1.3.34 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)
-

1.3.35 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 nonetheless. 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!

1.3.36 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 core dumps.
- 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 core dump proving the problem.

1.3.37 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 zone transfers. 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](#) 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.10](#).
- 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 file name. Thanks to Joris Vandalon for noticing this.
- The zone parser was more case-sensitive than BIND, refusing to accept 'in' as well as 'IN'. Thanks to Joris Vandalon for noticing this.

1.3.38 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](#), 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 16](#) 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](#).

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 nameservers 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.

1.3.39 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 17 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](#) 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.3.
- A recursing nameserver! See Chapter 17.
- 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.
- Zone parser 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).

1.3.40 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 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 zone files 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.7.2](#).

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 PostgreSQL backend as gpgsql2, all parameters will have gpgsql2 as a prefix, for example **gpgsql2-dbname**. If launched as gpgsql, 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.
 - Accessibility, 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 immediately.
- 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.7.2](#). This breaks compatibility 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.

1.3.41 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 webservice (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 occurred (observed by Paul Wouters)
- Documentation bug, pgmysql != mypgsql

Documentation:

- Documented the 'random backend', see Section [A.2](#).
- 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.9](#). 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 on every database under the sun.

1.3.42 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-existent variables.
- IBM Universal Database DB2 backend now included in source distribution (untested!)
- Oracle backend now included in source distribution (slightly 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
- `.deb` files 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).

1.3.43 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.

1.3.44 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.
-

1.3.45 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 to 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.

1.3.46 Version 2.6.1

Quick fix release for a big cache problem.

1.3.47 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 benchmarking 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>. 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 Technologies for spotting this bug.
- zone2sql did not parse Verisign zone files 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'.

1.3.48 Version 2.5.1

Brown paper bag 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>. See also Appendix C.

And some small changes:

- Added support for RFC 2308 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**. 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 counts** command outputs statistics on what is in the cache, which is useful to help optimize your caching strategy.

1.3.49 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 gmssql 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 **ttls** 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 sped 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 file descriptors 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.

1.3.50 Version 2.4

Developers: this version is compatible with the pdns-2.1 development kit, available on <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 PostgreSQL, 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.
- Webservers 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.

1.3.51 Version 2.3

Developers: this version is compatible with the pdns-2.1 development kit, available on <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.3.

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 indeterminate 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.

1.3.52 Version 2.2

Developers: this version is compatible with the pdns-2.1 development kit, available on <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 20.

1.3.53 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>. 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 overwrite previous configuration files. Thanks to Jorn Ekkelenkamp of Hubris/ISP Services for reporting this.
- TCP recursion sent out overly large responses due to a byte order 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 were not implemented because they are a performance penalty. To turn these on, enable **wildcard-uri** 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 16.

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.4.
- 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.

1.3.54 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 command line. 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.

1.3.55 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.
-

1.3.56 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 zone transfer 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.

1.3.57 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 (635). 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.
- Zone file parser was not compatible with bind when \$INCLUDING non-absolute file names. Thanks to Jeff Miller for working out how this should work.
- Bind configuration parser was not compatible with bind when including non-absolute file names. 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.

1.3.58 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 code base 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.8.
- IBM DB2 Universal Database backend available for Linux. See Section A.6.
- Zone2sql now understands \$INCLUDE. Thanks to Amaze Internet for nagging about this
- The SOA Minimum 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 semantic consequences 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 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](#).

1.3.59 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 suffixes in zone files, ie 2H instead of 7200 (thanks Henk Wevers)

1.3.60 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 [18](#) 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 20.
- 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 Rijdsdijk for suggesting saner defaults.
- Infrastructure for 'also-notify' statements added.

1.3.61 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 edited under operating systems 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 18.2.1.
- Recursing backend can now live on a non-standard (!=53) port. See Chapter 16.
- 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.
-

1.3.62 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 16. 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 18.2
- All record types now documented, see Chapter 22.

1.3.62.1 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

1.3.62.2 Missing features

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

- gmysqlbackend, oraclebackend
-

1.3.63 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 file names 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 PostgreSQL backend
- zone2sql is now documented.

1.3.63.1 Known bugs

Wildcard CNAMEs do not work as they do with bind.

1.3.63.2 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.

1.3.64 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
- File names 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.
-

1.3.64.1 Known bugs

FreeBSD version does not stop when requested to do so.

Wildcard CNAMEs do not work as they do with bind.

1.3.64.2 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.

1.3.65 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.conf files 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.

1.3.65.1 Known bugs

FreeBSD version does not stop when requested to do so.

Wildcard CNAMEs do not work as they do with bind.

1.3.65.2 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.

1.3.66 Version 1.99.4 Early Access Prerelease

A lot of new named.conf files 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.

1.3.66.1 Known bugs

FreeBSD version does not stop when requested to do so.

Zone2sql refuses named.conf files with less than 100 domains.

Wildcard CNAMEs do not work as they do with bind.

1.3.66.2 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.

1.3.67 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.conf files 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 zone files, 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.
- Webservice 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 wire speed (many megabits/s).

1.3.67.1 Known bugs

FreeBSD version does not stop when requested to do so.

BindBackend cannot parse zones with \$GENERATE statements.

1.3.67.2 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.

1.3.68 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 crash (dump core) 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.

1.3.68.1 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 file descriptors.

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

1.3.68.2 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.

1.3.69 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.

1.3.69.1 Known bugs

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

1.3.69.2 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

If you have a security problem to report, please email us at both security@netherlabs.nl and ahu@ds9a.nl. Please do not mail security issues to public lists, nor file a ticket, unless we do not get back to you in a timely manner. We fully credit reporters of security issues, and respond quickly, but please allow us a reasonable timeframe to coordinate a response.

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.

As of the 9th of January 2012, no actual security problems with PowerDNS 2.9.22.5, 3.0.1, Recursor 3.1.7.2, 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 mailing lists.

Versions 2.9.22 and lower and 3.0 of the PowerDNS Authoritative Server were vulnerable to a temporary denial of service attack. For more detail, see Section 1.12.

Version 3.1.7.1 and earlier of the PowerDNS Recursor were vulnerable to a probably exploitable buffer overflow and a spoofing attack. For more detail, see Section 1.10 and Section 1.11.

Version 3.1.4 and earlier of the PowerDNS recursor were vulnerable to a spoofing attack. For more detail, see Section 1.7.

Version 3.1.3 and earlier of the PowerDNS recursor contain two security issues, both of which can lead to a denial of service, both of which can be triggered by remote users. One of the issues might lead be exploited and lead to a system compromise. For more detail, see Section 1.5 and Section 1.6.

Version 3.0 of the PowerDNS recursor contains a denial of service bug which can be exploited remotely. This bug, which we believe to only lead to a crash, has been fixed in 3.0.1. There are no guarantees however, so an upgrade from 3.0 is highly recommended.

All versions of PowerDNS before 2.9.21.1 do not respond to certain queries. This in itself is not a problem, but since the discovery by Dan Kaminsky of a new spoofing technique, this silence for queries PowerDNS considers invalid, within a valid domain, allows attackers more chances to feed *other* resolvers bad data.

All versions of PowerDNS before 2.9.18 contain the following two bugs, which only apply to installations running with the LDAP backend, or installations providing recursion to a limited range of IP addresses. If any of these apply to you, an upgrade is highly advised:

- The LDAP backend did not properly escape all queries, allowing it to fail and not answer questions. We have not investigated further risks involved, but we advise LDAP users to update as quickly as possible (Norbert Sendetzky, Jan de Groot)
- Questions from clients denied recursion could blank out answers to clients who are allowed recursion services, temporarily. Reported by Wilco Baan. This would've made it possible for outsiders to blank out a domain temporarily to your users. Luckily PowerDNS would send out SERVFAIL or Refused, and not a denial of a domain's existence.

All versions of PowerDNS before 2.9.17 are known to suffer from remote denial of service problems which can disrupt operation. Please upgrade to 2.9.17 as this page will only contain detailed security information from 2.9.17 onwards.

CVE	CVE-2006-4251
Date	13th of November 2006
Affects	PowerDNS Recursor versions 3.1.3 and earlier, on all operating systems.
Not affected	No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
Severity	Critical
Impact	Potential remote system compromise.
Exploit	As far as we know, no exploit is available as of 11th of November 2006.
Solution	Upgrade to PowerDNS Recursor 3.1.4, or apply the patches referred below and recompile
Workaround	Disable TCP access to the Recursor. This will have slight operational impact, but it is likely that this will not lead to meaningful degradation of service. Disabling access is best performed at packet level, either by configuring a firewall, or instructing the host operating system to drop TCP connections to port 53. Additionally, exposure can be limited by configuring the allow-from setting so only trusted users can query your nameserver.

Table 1.1: PowerDNS Security Advisory

1.5 PowerDNS Security Advisory 2006-01: Malformed TCP queries can lead to a buffer overflow which might be exploitable

PowerDNS Recursor 3.1.3 and previous miscalculate the length of incoming TCP DNS queries, and will attempt to read up to 4 gigabytes of query into a 65535 byte buffer.

We have not verified if this problem might actually lead to a system compromise, but are acting on the assumption that it might.

For distributors, a minimal patch is available on [the PowerDNS wiki](#). Additionally, those shipping very old versions of the PowerDNS Recursor might benefit from this [patch](#).

The impact of these and other security problems can be lessened by considering the advice in [Chapter 7](#).

1.6 PowerDNS Security Advisory 2006-02: Zero second CNAME TTLs can make PowerDNS exhaust allocated stack space, and crash

PowerDNS would recurse endlessly on encountering a CNAME loop consisting entirely of zero second CNAME records, eventually exceeding resources and crashing.

1.7 PowerDNS Security Advisory 2008-01: System random generator can be predicted, leading to the potential to 'spoof' PowerDNS Recursor

We would like to thank Amit Klein of Trusteer for bringing a serious vulnerability to our attention which would enable a smart attacker to 'spoof' previous versions of the PowerDNS Recursor into accepting possibly malicious data.

Details can be found on [this Trusteer page](#).

This security problem was announced in [this email message](#).

It is recommended that all users of the PowerDNS Recursor upgrade to 3.1.5 as soon as practicable, while we simultaneously note that busy servers are less susceptible to the attack, but not immune.

CVE	CVE-2006-4252
Date	13th of November 2006
Affects	PowerDNS Recursor versions 3.1.3 and earlier, on all operating systems.
Not affected	No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
Severity	Moderate
Impact	Denial of service
Exploit	This problem can be triggered by sending queries for specifically configured domains
Solution	Upgrade to PowerDNS Recursor 3.1.4, or apply commit 919 .
Workaround	None known. Exposure can be limited by configuring the allow-from setting so only trusted users can query your nameserver.

Table 1.2: PowerDNS Security Advisory

CVE	Not yet assigned
Date	31st of March 2008
Affects	PowerDNS Recursor versions 3.1.4 and earlier, on most operating systems
Not affected	No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
Severity	Moderate
Impact	Data manipulation; client redirection
Exploit	This problem can be triggered by sending queries for specifically configured domains, sending spoofed answer packets immediately afterwards.
Solution	Upgrade to PowerDNS Recursor 3.1.5, or apply changesets 1159, 1160 and 1164 .
Workaround	None known. Exposure can be limited by configuring the allow-from setting so only trusted users can query your nameserver.

Table 1.3: PowerDNS Security Advisory

The vulnerability is present on all operating systems where the behaviour of the `libc random()` function can be predicted based on its past output. This includes at least all known versions of Linux, as well as Microsoft Windows, and probably FreeBSD and Solaris.

The magnitude of this vulnerability depends on internal details of the system `random()` generator. For Linux, the mathematics of the random generator are complex, but well understood and Amit Klein has written and published a proof of concept that can successfully predict its output after uninterrupted observation of 40-50 DNS queries.

Because the observation needs to be uninterrupted, busy PowerDNS Recursor instances are harder to subvert - other data is highly likely to be interleaved with traffic generated by an attacker.

Nevertheless, operators are urged to update at their earliest convenience.

1.8 PowerDNS Security Advisory 2008-02: By not responding to certain queries, domains become easier to spoof

CVE	CVE-2008-3337
Date	6th of August 2008
Affects	PowerDNS Authoritative Server 2.9.21 and earlier
Not affected	No versions of the PowerDNS Recursor (' <code>pdns_recursor</code> ') are affected.
Severity	Moderate
Impact	Data manipulation; client redirection
Exploit	Domains with servers that drop certain queries can be spoofed using simpler measures than would usually be required
Solution	Upgrade to PowerDNS Authoritative Server 2.9.21.1, or apply commit 1239 .
Workaround	None known.

Table 1.4: PowerDNS Security Advisory

Brian J. Dowling of Simplicity Communications has discovered a security implication of the previous PowerDNS behaviour to drop queries it considers malformed. We are grateful that Brian notified us quickly about this problem.

The implication is that while the PowerDNS Authoritative server itself does not face a security risk because of dropping these malformed queries, other resolving nameservers run a higher risk of accepting spoofed answers for domains being hosted by PowerDNS Authoritative Servers before 2.9.21.1.

While the dropping of queries does not aid sophisticated spoofing attempts, it does facilitate simpler attacks.

1.9 PowerDNS Security Advisory 2008-02: Some PowerDNS Configurations can be forced to restart remotely

Daniel Drown discovered that his PowerDNS 2.9.21.1 installation crashed on receiving a HINFO CH query. In his enthusiasm, he shared his discovery with the world, forcing a rapid over the weekend release cycle.

While we thank Daniel for his discovery, please study our security policy as outlined in Section 1.4 before making vulnerabilities public.

It is believed that this issue only impacts PowerDNS Authoritative Servers operating with '`distributor-threads=1`', but even on other configurations a database reconnect occurs on receiving a CH HINFO query.

CVE	Not yet assigned
Date	18th of November 2008
Affects	PowerDNS Authoritative Server 2.9.21.1 and earlier
Not affected	No versions of the PowerDNS Recursor ('pdns_recursor') are affected. Versions not running in single threaded mode ('distributor-threads=1') are probably not affected.
Severity	Moderate
Impact	Denial of Service
Exploit	Send PowerDNS an CH HINFO query.
Solution	Upgrade to PowerDNS Authoritative Server 2.9.21.2, or wait for 2.9.22.
Workaround	Remove 'distributor-threads=1' if this is set.

Table 1.5: PowerDNS Security Advisory

1.10 PowerDNS Security Advisory 2010-01: PowerDNS Recursor up to and including 3.1.7.1 can be brought down and probably exploited

CVE	CVE-2009-4009
Date	6th of January 2010
Affects	PowerDNS Recursor 3.1.7.1 and earlier
Not affected	No versions of the PowerDNS Authoritative ('pdns_server') are affected.
Severity	Critical
Impact	Denial of Service, possible full system compromise
Exploit	Withheld
Solution	Upgrade to PowerDNS Recursor 3.1.7.2 or higher
Workaround	None. The risk of exploitation or denial of service can be decreased slightly by using the 'allow-from' setting to only provide service to known users. The risk of a full system compromise can be reduced by running with a suitable reduced privilege user and group settings, and possibly chroot environment.

Table 1.6: PowerDNS Security Advisory

Using specially crafted packets, it is possible to force a buffer overflow in the PowerDNS Recursor, leading to a crash.

This vulnerability was discovered by a third party that (for now) prefers not to be named. PowerDNS is very grateful however for their help in improving PowerDNS security.

1.11 PowerDNS Security Advisory 2010-02: PowerDNS Recursor up to and including 3.1.7.1 can be spoofed into accepting bogus data

Using specially crafted zones, it is possible to fool the PowerDNS Recursor into accepting bogus data. This data might be harmful to your users. An attacker would be able to divert data from, say, bigbank.com to an IP address of his choosing.

This vulnerability was discovered by a third party that (for now) prefers not to be named. PowerDNS is very grateful however for their help in improving PowerDNS security.

CVE	CVE-2009-4010
Date	6th of January 2010
Affects	PowerDNS Recursor 3.1.7.1 and earlier
Not affected	No versions of the PowerDNS Authoritative ('pdns_server') are affected.
Severity	High
Impact	Using smart techniques, it is possible to fool the PowerDNS Recursor into accepting unauthorized data
Exploit	Withheld
Solution	Upgrade to PowerDNS Recursor 3.1.7.2 or higher
Workaround	None.

Table 1.7: PowerDNS Security Advisory

1.12 PowerDNS Security Advisory 2012-01: PowerDNS Authoritative Server can be caused to generate a traffic loop

CVE	CVE-2012-0206
Date	10th of January 2012
Credit	Ray Morris of BetterCGI.com .
Affects	Most PowerDNS Authoritative Server versions < 3.0.1 (with the exception of 2.9.22.5 and 2.9.22.6)
Not affected	No versions of the PowerDNS Recursor ('pdns_recursor') are affected.
Severity	High
Impact	Using well crafted UDP packets, one or more PowerDNS servers could be made to enter a tight packet loop, causing temporary denial of service
Exploit	Proof of concept
Risk of system compromise	No
Solution	Upgrade to PowerDNS Authoritative Server 2.9.22.5 or 3.0.1
Workaround	Several, the easiest is setting: cache-ttl=0, which does have a performance impact. Please see below.

Table 1.8: PowerDNS Security Advisory

Affected versions of the PowerDNS Authoritative Server can be made to respond to DNS responses, thus enabling an attacker to setup a packet loop between two PowerDNS servers, perpetually answering each other's answers. In some scenarios, a server could also be made to talk to itself, achieving the same effect.

If enough bouncing traffic is generated, this will overwhelm the server or network and disrupt service.

As a workaround, if upgrading to a non-affected version is not possible, several options are available. The issue is caused by the packet-cache, which can be disabled by setting 'cache-ttl=0', although this does incur a performance penalty. This can be partially addressed by raising the query-cache-ttl to a (far) higher value.

Alternatively, on Linux systems with a working iptables setup, 'responses' sent to the PowerDNS Authoritative Server 'question' address can be blocked by issuing:

```
iptables -I INPUT -p udp --dst $AUTHIP --dport 53 \! -f -m u32 --u32 "0>>22&0x3C@8 ←
>>15&0x01=1" -j DROP
```

If this command is used on a router or firewall, substitute FORWARD for INPUT.

To solve this issue, we recommend upgrading to the latest packages available for your system. Tarballs and new static builds (32/64bit, RPM/DEB) of 2.9.22.5 and 3.0.1 have been uploaded to [our download site](#). Kees Monshouwer has provided updated CentOS/RHEL packages in [his repository](#). Debian, Fedora and SuSE should have packages available shortly after this announcement.

For those running custom PowerDNS versions, just applying this patch may be easier:

```
--- pdns/common_startup.cc      (revision 2326)
+++ pdns/common_startup.cc      (working copy)
@@ -253,7 +253,9 @@
     numreceived4++;
     else
     numreceived6++;
-
+   if(P->d.qr)
+       continue;
+
   S.ringAccount("queries", P->qdomain+"/"+P->qtype.getName());
   S.ringAccount("remotes", P->getRemote());
   if(logDNSQueries) {
```

It should apply cleanly to 3.0 and with little trouble to several older releases, including 2.9.22 and 2.9.21.

This bug resurfaced because over time, the check for 'not responding to responses' moved to the wrong place, allowing certain responses to be processed anyhow.

We would like to thank Ray Morris of [BetterCGI.com](#) for bringing this issue to our attention and Aki Tuomi for helping us reproduce the problem.

1.13 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 mailing list

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
```



Warning

As of 2.9.21, the BIND backend no longer features the 'bind-example-zones' command.

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, is

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

**Warning**

As of 3.0 and up, Windows support is untested and probably does not work at all.

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.

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`).

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 DSN, 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.8](#).

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 command line 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 command line 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

Basic setup: configuring 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=pdns
```

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.3](#). If you use the 'mysql' backend things will only appear to work. (The 'mysql' backend was removed in version 3.1).

Now start PDNS using the monitor command:

```
# /etc/init.d/pdns monitor
(...)
15:31:30 About to create 3 backend threads
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdns'
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdns'
15:31:30 [gMySQLbackend] Failed to connect to database: Error: Unknown database 'pdns'
```

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 Example: 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(128) 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)  
) Engine=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(10) DEFAULT NULL,  
  content     VARCHAR(64000) DEFAULT NULL,  
  ttl         INT DEFAULT NULL,  
  prio        INT DEFAULT NULL,  
  change_date INT DEFAULT NULL,  
  primary key (id)  
) Engine=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  
) Engine=InnoDB;
```

```
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 response ←
```

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);
```



Warning

Host names and the MNAME of a SOA records are NEVER terminated with a '.' in PowerDNS storage! If a trailing '.' is present it will inevitably cause problems, problems that may be hard to debug.

If we now query 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 load balancing, 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 Section [A.1](#).

Chapter 6

Logging & Monitoring Authoritative Server 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>

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 priorities - 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 disk space reasons, it is advised to audit your syslog.conf for statements also logging PDNS activities. Many syslog.conf's 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 file names not starting with a '-' - indicating a synchronous log, which hurts performance.

Be aware that syslog by default logs messages at the configured priority and higher! To log only info messages, use **local0.=info**.

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.

For additional information on PowerDNS security, PowerDNS security incidents and PowerDNS security policy, see Section 1.4.

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.



Warning

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. When running with master or slave support, be aware that many operating systems need access to specific libraries (often `/lib/libnss*`) in order to support resolution of domain names! You can also hardlink these. In addition, make sure that `/dev/log` is available from within the chroot. Logging will silently fail over time otherwise (on logrotate).

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 emphatically 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

Authoritative Server Performance

9.1 General advice

In general, best performance is achieved on recent Linux 2.6 kernels and using MySQL, although many of the largest PowerDNS installations are based on PostgreSQL. FreeBSD appears to achieve lower packet rates both for the PowerDNS recursor as for the authoritative nameserver, this is still being investigated. No comparative measurements have been done for Solaris installations.

On Linux, make sure to read Section [9.2](#).

Database servers can require configuration to achieve decent performance. It is especially worth noting that several vendors ship PostgreSQL with a slow default configuration.

9.2 Native Posix Thread Library vs LinuxThreads

To get the best performance under Linux, especially on SMP systems, the use of NPTL is advised. The difference in performance can be over a factor of ten in some circumstances.

NPTL is the default library on modern Linux distributions, so there is generally not a problem, except if you use a statically compiled version that, for portability reasons, defaults to LinuxThreads. This includes all .deb and .rpm files provided by us up to and including 2.9.18.

When running a PowerDNS-provided static binary of 2.9.18 or lower, it may make sense to recompile, or to upgrade to a newer version, if available. When recompiling, be sure to use a supported compiler, like g++ >3.2. You might also consider moving to a distribution supplied version.

A good indication that your installation might benefit from such an upgrade is to watch the 'cs' count in the output of `vmstat 1`. If this is very high (> 10000), you are suffering from a LinuxThreads performance problem called 'overspin'.

Thanks are due to L. Bunt Jackson who noted the static compilation problem in an article in Dr. Dobb's Journal.

9.3 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 20 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.

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.

9.3.1 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`

9.3.2 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-cache-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 PowerDNS

Before migrating to PowerDNS a few things should be considered.

PowerDNS does not operate as a 'slave' or 'master' server with all backends Only the Generic SQL, OpenDBX and BIND backends have the ability to act as master or slave.

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

Additionally, the PowerDNS source comes with a number of diagnostic tools, which can be helpful in verifying proper PowerDNS operation, versus incumbent nameservers. See Chapter 25 for more details.

10.1 Zone2sql

Zone2sql parses Bind named.conf files and zone files 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>t1
```

--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.

- 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

Notes on upgrading

11.1 From PowerDNS Authoritative Server 2.9.x to 3.0

The 3.0 release of the PowerDNS Authoritative Server is significantly different from previous 2.9.x versions. This section lists important things to be aware of.



Warning

Version 3.0 of the PowerDNS Authoritative Server is the biggest change in PowerDNS history. In some senses, this means that it behaves somewhat like a '1.0' version. We advise operators to carefully perform the upgrade process from 2.9.x, and if possible test on a copy of the database beforehand.

In addition, it may also be useful to have a support agreement in place during such upgrades. For first class and rapid support, please contact powerdns-support@netherlabs.nl, or see www.powerdns.com. Alternatively, the **PowerDNS Community** can be very helpful too.

With similar settings, version 3.0 will most likely use a lot more memory than 2.9. This is due to the new DNSSEC key & signature caches, but also because the database query cache will now store multiple row answers, which it did not do previously. Memory use can be brought down again by tuning the `cache-ttl` settings.

Performance may be up, or it may be down. We appreciate that this is spotty guidance, but depending on your setup, lookups may be a lot faster or a lot slower. The improved database cache may prove to be a big benefit, and improve performance dramatically. This could be offset by a near duplication of database queries needed because of more strict interpretation of DNS standards.

PowerDNS Authoritative Server 3.0 contains a completely renewed implementation of the core DNS 'Algorithm', loosely specified in RFC 1034. As stated above, our new implementation is a lot closer to the original standard. This may mean that version 3.0 may interpret the contents of your database differently from how 2.9.x interpreted them. For fully standards confirming zones, there should not be a problem, but if zones were misconfigured (no SOA record, for example), things will be different.

When compiling version 3.0, there are now more dependencies than there used to be. Whereas previously, only Boost header files were needed, PowerDNS now needs a number of Boost libraries to be installed (like `boost-program-options`, `boost-serialization`). In addition, for now Lua 5.1 is a dependency.

PowerDNS Authoritative Server 3.0 comes with DNSSEC support, but this has required big changes to database schemas. Each backend lists the changes required. To facilitate a smooth upgrade, the old, non-DNSSEC schema is used by default. Features like per-domain metadata, TSIG and DNSSEC itself however need the new schema. Consult your backend documentation for the correct 'alter table' statements. Afterwards, set the relevant '-dnssec' setting for your backend (for example: `gmysql-dnssec`).

In version 3.0, "Fancy Records", like URL, CURL and MBOXFW are no longer supported. Support may come back in later versions. In addition, the LDAP Backend has moved to 'unmaintained' status.

11.1.1 Frequently Asked Questions about 3.0

Q: Can 2.9.x versions read the 3.0 DNSSEC database schema? A: Yes, every database can be altered to the new schema without impact on 2.9. The new fields and tables are ignored.

Q: Can 3.x versions read the 2.9 pre-DNSSEC database schema? A: Yes, as long as the relevant '-dnssec' setting is not enabled. These settings are typically called 'gmysql-dnssec', 'gpgsql-dnssec', 'sqlite3-dnssec'. If this setting IS enabled, 3.x expects the new schema to be in place.

Q: If I run 3.0 with the new schema, and I have set '-dnssec', do I need to rectify my zones? A: Yes. If the '-dnssec' setting is enabled, PowerDNS expects the 'auth' field to be filled out correctly. When slaving zones this happens automatically. For other zones, run 'pdnssec rectify-zone zonename'. Even if a zone is not DNSSEC secured, as long as the new schema is in place, the zone must be rectified (or at least have the 'auth' field set correctly).

Q: I want to fill out the 'auth' and 'ordername' fields directly, how do I do this? A: The 'auth' field should be '1' or 'true' for all records that are within your zone. For a zone without delegations, this means 'auth' should always be set. If you have delegations, both the NS records for that delegation and possible glue records for it should not have 'auth' set.

For more details on 'auth' and 'ordername', please see Section [12.8.5](#).

Q: If I don't update to the new DNSSEC schema, will 3.0 give identical answers as 2.9.x? A: Not always. The core DNS logic of 3.0 was changed, so even if no changes are made to the database, you may get different answers. This might happen for zones without SOA records for example, which used to (more or less) work. An upgrade from 2.9.x to 3.0 should always be monitored carefully.

11.2 From PowerDNS Authoritative Server 3.0 to 3.1

Note

If you are coming from 2.9.x, please also read Section [11.1](#).

PowerDNS 3.1 introduces native SQLite3 support for storing key material for DNSSEC in the bindbackend. With this change, support for bind+gsql-setups ('hybrid mode') has been dropped. If you were using this mode, you will need to switch to bind-dnssec-db and migrate your keying material.

There have been changes to the SQL schemas for the generic backends.

For MySQL:

```
mysql> ALTER TABLE records MODIFY content VARCHAR(64000);
mysql> ALTER TABLE tsigkeys MODIFY algorithm VARCHAR(50);
```

The definition of 'auth' and 'ordername' in backends has changed slightly, see Section [12.8.5](#).

PowerDNS 3.0 and 3.1 will only fetch DNSSEC metadata and key material from the first DNSSEC-capable backend in the launch line. In 3.1, the bindbackend supports DNSSEC storage. This means that setups using launch=bind,gsqlite3 or launch=gsqlite3,bind may break. Please tread carefully!

Chapter 12

Serving authoritative DNSSEC data

(only available in PowerDNS 3.0 and beyond, not yet available in the PowerDNS Recursor)

PowerDNS contains support for DNSSEC, enabling the easy serving of DNSSEC secured data, with minimal administrative overhead.

In PowerDNSSEC, DNS and signatures and keys are (usually) treated as separate entities. The domain & record storage is thus almost completely devoid of DNSSEC record types.

Instead, keying material is stored separately, allowing operators to focus on the already complicated task of keeping DNS data correct. In practice, DNSSEC related material is often stored within the same database, but within separate tables.

If a DNSSEC configuration is found for a domain, the PowerDNS daemon will provide keys, signatures and (hashed) denials of existence automatically.

As an example, securing an existing zone can be as simple as:

```
$ pdnssec secure-zone powerdnssec.org
$ pdnssec rectify-zone powerdnssec.org
```

Alternatively, PowerDNS can serve pre-signed zones, without knowledge of private keys.

12.1 A brief introduction to DNSSEC

DNSSEC is a complicated subject, but it is not required to know all the ins and outs of this protocol to be able to use PowerDNSSEC. In this section, we explain the core concepts that are needed to operate a PowerDNSSEC installation.

Zone material is enhanced with signatures using 'keys'. Such a signature (called an RRSIG) is a cryptographic guarantee that the data served is the original data. DNSSEC keys are asymmetric (RSA, DSA or GOST), the public part is published over DNS and is called a DNSKEY record, and is used for verification. The private part is used for signing and is never published.

To make sure that the internet knows that the key that is used for signing is the authentic key, confirmation can be gotten from the parent zone. This means that to become operational, a zone operator will have to publish a representation of the signing key to the parent zone, often a ccTLD or a gTLD. This representation is called a DS record, and is a shorter (hashed) version of the DNSKEY.

Once the parent zone has the DS, and the zone is signed with the DNSSEC key, we are done in theory.

However, for a variety of reasons, most DNSSEC operations run with another layer of keys. The so called 'Key Signing Key' is sent to the parent zone, and this Key Signing Key is used to sign a new set of keys called the Zone Signing Keys.

This setup allows us to change our keys without having to tell the zone operator about it.

A final challenge is how to DNSSEC sign the answer 'no such domain'. In the language of DNS, the way to say 'there is no such domain' (NXDOMAIN) or there is no such record type is to send an empty answer. Such empty answers are universal, and can't be signed.

In DNSSEC parlance we therefore sign a record that says 'there are no domains between A.powerdnssec.org and C.powerdnssec.org'. This securely tells the world that B.powerdnssec.org does not exist. This solution is called NSEC, and is simple but has downsides - it also tells the world exactly which records DO exist.

So alternatively, we can say that if a certain mathematical operation (an 'iterated salted hash') is performed on a question, that no valid answers exist that have as outcome of this operation an answer between two very large numbers. This leads to the same 'proof of non-existence'. This solution is called NSEC3.

A PowerDNSSEC zone can either be operated in NSEC or in one of two NSEC3 modes ('inclusive' and 'narrow').

12.2 Profile, Supported Algorithms, Record Types & Modes of operation

PowerDNSSEC aims to serve unexciting, standards compliant, DNSSEC information. One goal is to have relevant parts of our output be identical or equivalent to important fellow-traveller software like NLNetLabs' NSD.

Particularly, if a PowerDNSSEC secured zone is transferred via AXFR, it should be able to contain the same records as when that zone was signed using 'ldns-signzone' using the same keys and settings.

PowerDNS supports serving pre-signed zones, as well as online ('live') signed operations. In the last case, Signature Rollover and Key Maintenance are fully managed by PowerDNS.

In addition to the above, PowerDNSSEC also supports modes of operation which may not have an equivalent in other pieces of software, for example NSEC3-narrow mode.

PowerDNSSEC supports:

- NSEC
- NSEC3
- NSEC3-narrow
- DS (digest type 1, 2, 3 and provisional point 4)
- RSASHA1 (algorithm 5, algorithm 7)
- RSASHA256 (algorithm 8)
- RSASHA512 (algorithm 10)
- ECC-GOST (algorithm 12)
- ECDSA (no codepoints assigned, provisional 13 and 14)

This corresponds to:

- RFC 4033: DNS Security Introduction and Requirements
 - RFC 4034: Resource Records for the DNS Security Extensions, Protocol Modifications for the DNS Security Extensions
 - RFC 4035: Protocol Modifications for the DNS Security Extensions
 - RFC 4509: Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)
 - RFC 5155: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence
 - RFC 5702: Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC
 - RFC 5933: Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC
 - draft-ietf-dnsext-ecdsa: Elliptic Curve DSA for DNSSEC
-

12.2.1 DNSSEC: live-signed vs orthodox 'pre-signed' mode

Traditionally, DNSSEC signatures have been added to unsigned zones, and then this signed zone could be served by any DNSSEC capable authoritative server. PowerDNS supports this mode fully.

In addition, PowerDNS supports taking care of the signing itself, in which case PowerDNS operates differently from most tutorials and handbooks. This mode is easier however.

For relevant tradeoffs, please see Section 12.9 and Section 12.10.

12.3 Migration

This chapter discusses various migration strategies, from existing PowerDNS setups, from existing unsigned installations and finally from previous non-PowerDNS DNSSEC deployments.

12.3.1 From an existing PowerDNS installation

To migrate an existing database-backed PowerDNS installation, a few changes must be made to the database schema. First, the records table gains two new fields: 'auth' and 'ordername'. Some data in a zone, like glue records, should not be signed, and this is signified by setting 'auth' to 0.



Warning

Once the database schema has been updated, and the relevant 'gsqldb-dnssec' switch has been set, stricter rules apply for filling out the database! The short version is: run `pdnssec rectify-zone` on all zones, even those not secured with DNSSEC!

Additionally, NSEC and NSEC3 in non-narrow mode require ordering data in order to perform (hashed) denial of existence. The 'ordername' field is used for this purpose.

Finally, two new tables are needed. DNSSEC keying material is stored in the 'cryptokeys' table (in a portable standard format). Domain metadata is stored in the 'domainmetadata' table. This includes NSEC3 settings.

Once the database schema has been changed for DNSSEC usage (see the relevant backend chapters or [the PowerDNSSEC wiki](#) for the update statements), the 'pdnssec' tool can be used to fill out keying details, and 'rectify' the auth and ordername fields.

In short, `'pdnssec secure-zone powerdnssec.org ; pdnssec rectify-zone powerdnssec.org'` will deliver a correctly NSEC signed zone.

In addition, so will the 'zone2sql' import tool when run with the '--dnssec' flag.

12.3.2 From existing non-DNSSEC non-PowerDNS setups

TBD

12.3.3 From existing DNSSEC non-PowerDNS setups, pre-signed

Industry standard signed zones can be served natively by PowerDNS, without changes. In such cases, signing happens externally to PowerDNS, possibly via OpenDNSSEC, Idns-sign or dnssec-sign.

PowerDNS needs to know if a zone should receive DNSSEC processing. To configure, run `'pdnssec set-presigned zone'`.



Warning

Right now, you will also need to configure NSEC(3) settings for pre-signed zones using `'pdnssec set-nsec3'`. Default is NSEC, in which case no further configuration is necessary.

12.3.4 From existing DNSSEC non-PowerDNS setups, live signing

The 'pdnssec' tool features the option to import zone keys in the industry standard private key format, version 1.2. To import an existing KSK, use 'pdnssec import-zone-key zonename filename KSK', replace KSK by ZSK for a Zone Signing Key.

If all keys are imported using this tool, a zone will serve mostly identical records to before, with the important change that the RRSIG inception dates will be different.

Note

Within PowerDNS, the 'algorithm' for RSASHA1 keys is modulated based on the NSEC3 setting. So if an algorithm=7 key is imported in a zone with no configured NSEC3, it will appear as algorithm 5!

12.4 Records, Keys, signatures, hashes within PowerDNSSEC in online signing mode

Within PowerDNSSEC live signing, keys are stored separately from the zone records. Zone data are only combined with signatures and keys when requests come in over the internet.

Each zone can have a number of keys associated with it, with varying key lengths. Typically 1 or at most 2 of these keys are employed as actual Zone Signing Keys (ZSKs). During normal operations, this means that only 1 ZSK is 'active', and the other is passive.

Should it be desired to 'roll over' to a new key, both keys can temporarily be active (and used for signing), and after a while the old key can be inactivated. Subsequently it can be removed.

As elucidated above, there are several ways in which DNSSEC can deny the existence of a record, and this setting too is stored away from zone records, and lives with the DNSSEC keying material.

In order to facilitate interoperability with existing technologies, PowerDNSSEC keys can be imported and exported in industry standard formats.

Keys and hashes are configured using the 'pdnssec' tool, which is described next.

12.4.1 (Hashed) Denial of Existence

PowerDNS supports unhashed secure denial of existence using NSEC records. These are generated with the help of the (database) backend, which needs to be able to supply the 'previous' and 'next' records in canonical ordering.

The Generic SQL Backends have fields that allow them to supply these relative record names.

In addition, hashed secure denial of existence is supported using NSEC3 records, in two modes, one with help from the database, the other with the help of some additional calculations.

NSEC3 in 'broad' or 'inclusive' mode works with the aid of the backend, where the backend should be able to supply the previous and next domain names in hashed order.

NSEC3 in 'narrow' mode uses additional hashing calculations to provide hashed secure denial of existence 'on the fly', without further involving the database.

12.4.2 Signatures

In PowerDNS live signing mode, signatures, as served through RRSIG records, are calculated on the fly, and heavily cached. All CPU cores are used for the calculation.

RRSIGs have a validity period, in PowerDNS by default this period starts at most a week in the past, and continues at least a week into the future.

Precisely speaking, the time period used is always from the start of the previous Thursday until the Thursday two weeks later. This two-week interval jumps with one-week increments every Thursday.

Note

Why Thursday? POSIX-based operating systems count the time since GMT midnight January 1st of 1970, which was a Thursday. PowerDNS inception/expiration times are generated based on an integral number of weeks having passed since the start of the 'epoch'.

12.5 'pdnssec' for PowerDNSSEC command & control

'pdnssec' is a powerful command that is the operator-friendly gateway into PowerDNSSEC configuration. Behind the scenes, 'pdnssec' manipulates a PowerDNS backend database, which also means that for many databases, 'pdnssec' can be run remotely, and can configure key material on different servers.

The following pdnssec commands are available:

- activate-zone-key** **ZONE KEY-ID** Activate a key with id KEY-ID within a zone called ZONE.
- add-zone-key** **ZONE [ksklzsk] [bits] [rsasha1|rsasha256|rsasha512|gost|ecdsa256|ecdsa384]** Create a new key for zone ZONE, and make it a KSK or a ZSK, with the specified algorithm.
- check-zone** **ZONE** Check a zone for DNSSEC correctness. Main goal is to check if the auth flag is set correctly.
- check-all-zones** Check all zones for DNSSEC correctness. Added in 3.1.
- deactivate-zone-key** **ZONE KEY-ID** Deactivate a key with id KEY-ID within a zone called ZONE.
- export-zone-dnskey** **ZONE KEY-ID** Export to standard output DNSKEY and DS of key with key id KEY-ID within zone called ZONE.
- export-zone-key** **ZONE KEY-ID** Export to standard output full (private) key with key id KEY-ID within zone called ZONE. The format used is compatible with BIND and NSD/LDNS.
- hash-zone-record** **ZONE RECORDNAME** This convenience command hashes the name 'recordname' according to the NSEC3 settings of ZONE. Refuses to hash for zones with no NSEC3 settings.
- import-zone-key** **ZONE filename [ksklzsk]** Import from 'filename' a full (private) key for zone called ZONE. The format used is compatible with BIND and NSD/LDNS. KSK or ZSK specifies the flags this key should have on import.
- import-zone-key-pem** **ZONE filename algorithm [ksklzsk]** Import from 'filename' a full (private) key in PEM format for zone called ZONE, and assign it an algorithm number. KSK or ZSK specifies the flags this key should have on import. The format used is compatible with 'openssl genrsa', which is also called PEM.
- rectify-zone** **ZONE [ZONE ..]** Calculates the 'ordname' and 'auth' fields for a zone called ZONE so they comply with DNSSEC settings. Can be used to fix up migrated data. Can always safely be run, it does no harm. Multiple zones can be supplied.
- rectify-all-zones** Do a rectify-zone for all the zones. Be careful when running this. Only bind and gmysql backends are supported. Added in 3.1.
- remove-zone-key** **ZONE KEY-ID** Remove a key with id KEY-ID from a zone called ZONE.
- secure-zone** **ZONE** Configures a zone called ZONE with reasonable DNSSEC settings. You should manually run 'rectify-zone' afterwards.
- set-nsec3** **ZONE 'parameters' [narrow]** Sets NSEC3 parameters for this zone. A sample command line is: "pdnssec set-nsec3 powerdnssec.org '1 1 1 ab' narrow". The NSEC3 parameters must be quoted on the command line.

**Warning**

If running in RSASHA1 mode (algorithm 5 or 7), switching from NSEC to NSEC3 will require a DS update at the parent zone!

The NSEC3 fields are: 'algorithm flags iterations salt'. Both 'algorithm' and 'flags' should be 1 for PowerDNS operation.

set-presigned ZONE Switches zone to presigned operation, utilizing in-zone RRSIGs.

show-zone ZONE Shows all DNSSEC related settings of a zone called ZONE.

unset-nsec3 ZONE Converts a zone to NSEC operations.

**Warning**

If running in RSASHA1 mode (algorithm 5 or 7), switching from NSEC to NSEC3 will require a DS update at the parent zone!

unset-presigned ZONE Disables presigned operation for ZONE.

12.6 DNSSEC advice & precautions

DNSSEC is a major change in the way DNS works. Furthermore, there is a bewildering array of settings that can be configured.

It is well possible to configure DNSSEC in such a way that your domain will not operate reliably, or even, at all.

We advise operators to stick to the keying defaults of 'pdnssec secure-zone': RSASHA256 (algorithm 8), 1 Key Signing Key of 2048 bits, 1 active Zone Signing Key of 1024 bits, 1 passive Zone Signing Key of 1024 bits.

While the 'GOST' and 'ECDSA' algorithms are better choices in theory, not many DNSSEC resolvers can validate answers signed with such keys. Much the same goes for RSASHA512, except that it does not offer better performance either.

Note

GOST may be more widely available in Russia, because it might be mandatory to implement this regional standard there.

It is possible to operate a zone with different keying algorithms simultaneously, but it has also been observed that this is not reliable.

12.6.1 Packet sizes, fragments, TCP/IP service

DNSSEC answers contain (bulky) keying material and signatures, and are therefore a lot larger than regular DNS answers. Normal DNS responses almost always fit in the 'magical' 512 byte limit previously imposed on DNS.

In order to support DNSSEC, operators must make sure that their network allows for:

- >512 byte UDP packets on port 53
- Fragmented UDP packets
- ICMP packets related to fragmentation
- TCP queries on port 53
- EDNS0 queries/responses (filtered by some firewalls)

If any of the conditions outlined above is not met, DNSSEC service will suffer or be completely unavailable.

In addition, the larger your DNS answers, the more critical the above becomes. It is therefore advised not to provision too many keys, or keys that are unnecessarily large.

12.7 Operational instructions

In this chapter various DNSSEC transitions are discussed, and how to execute them within PowerDNSSEC.

12.7.1 Publishing a DS

To publish a DS to a parent zone, utilize 'pdnssec show-zone' and take the DS from its output, and transfer it securely to your parent zone.

12.7.2 ZSK rollover

```
.. pdnssec activate-zone-key ZONE next-key-id .. .. pdnssec deactivate-zone-key ZONE prev-key-id .. .. pdnssec remove-zone-key ZONE prev-key-id ..
```

12.7.3 KSK rollover

```
.. pdnssec show-zone ZONE and communicate duplicate DS .. .. pdnssec activate-zone-key ZONE next-key-id .. .. pdnssec deactivate-zone-key ZONE prev-key-id .. .. pdnssec remove-zone-key ZONE prev-key-id ..
```

12.7.4 Going insecure

```
.. pdnssec disable-dnssec ..
```

12.7.5 NSEC(3) change

```
.. pdnssec show-zone ZONE and communicate duplicate DS .. .. pdnssec set-nsec3 'parameters' ZONE
```

12.8 Modes of operation

PowerDNSSEC can operate in several modes. In the simplest situation, there is a single "SQL" database that contains, in separate tables, all domain data, keying material and other DNSSEC related settings.

This database is then replicated to all PowerDNS instances, which all serve identical records, keys and signatures.

In this mode of operation, care should be taken that the database replication occurs over a secure network, or over an encrypted connection. This is because keying material, if intercepted, could be used to counterfeit DNSSEC data using the original keys.

Such a single replicated database requires no further attention beyond monitoring already required during non-DNSSEC operations.

12.8.1 PowerDNSSEC Pre-signed records

In this mode, PowerDNS serves zones that already contain DNSSEC records. Such zones can either be slaved from a remote master, or can be signed using tools like OpenDNSSEC, ldns-signzone or dnssec-signzone.

12.8.2 PowerDNSSEC Front-signing

As a special feature, PowerDNSSEC can operate as a signing server which operates as a slave to an unsigned master.

In this way, if keying material is available for an unsigned zone that is retrieved from a master server, this keying material will be used when serving data from this zone.

As part of the zone retrieval, the equivalent of 'pdnssec rectify-zone' is run to make sure that all DNSSEC-related fields are set correctly.

12.8.3 PowerDNSSEC BIND-mode operation

Starting with PowerDNS 3.1, the bindbackend can manage keys in an SQLite3 database without launching a separate gsqlite3 backend.

To use this mode, add "bind-dnssec-db=/var/db/bind-dnssec-db.sqlite3" to pdns.conf, and run "pdnssec create-bind-db /var/db/bind-dnssec-db.sqlite3". Then, restart PowerDNS.

After this, you can use "pdnssec secure-zone" and all other pdnssec commands on your BIND zones without trouble.

12.8.4 PowerDNSSEC hybrid BIND-mode operation

**Warning**

This mode is only supported in 3.0 and 3.0.1! In 3.1 and up, the bindbackend always does its own key storage.

PowerDNS can also operate based on 'BIND'-style zone & configuration files. This 'bindbackend' has full knowledge of DNSSEC, but has no native way of storing keying material.

However, since PowerDNS supports operation with multiple simultaneous backends, this is not a problem.

In hybrid mode, keying material and zone records are stored in different backends. This allows for 'bindbackend' operation in full DNSSEC mode.

To benefit from this mode, include at least one database-based backend in the 'launch' statement. The Generic SQLite backend version 3 (gsqlite3) probably complements BIND mode best, since it does not require a database server process.

**Warning**

For now, it is necessary to execute a manual SQL 'insert' into the domains table of the backend hosting the keying material. This is needed to generate a zone-id for the relevant domain. Sample SQL statement: **insert into domains (name, type) values ('powerdnssec.org', 'NATIVE');**

12.8.5 Rules for filling out fields in database backends

Note

The BIND Backend automates all the steps outlined below, and does not need 'manual' help

For DNSSEC, two additional fields are important: 'auth' and 'ordname'. These fields are set correctly on an incoming zone transfer, and also by running 'pdnssec rectify-zone', or 'zone2sql' with the --dnssec flag.

The 'auth' field should be set to '1' for data for which the zone itself is authoritative, which includes the SOA record and its own NS records.

The 'auth' field should be 0 however for NS records which are used for delegation, and also for any glue (A, AAAA) records present for this purpose. Do note that the DS record for a secure delegation should be authoritative!

The 'ordname' field needs to be filled out depending on the NSEC/NSEC3 mode. When running in NSEC3 'Narrow' mode, the ordname field is ignored and best left empty. In NSEC mode, the ordname field should be NULL for any glue but filled in for delegation NS records and all authoritative records. In NSEC3 opt-out mode (the only NSEC3 mode PowerDNS currently supports), any non-authoritative records (as described for the 'auth' field) should have ordname set to NULL.

In 'NSEC' mode, it should contain the *relative* part of a domain name, in reverse order, with dots replaced by spaces. So 'www.uk.powerdnssec.org' in the 'powerdnssec.org' zone should have 'uk www' as its ordname.

In 'NSEC3' non-narrow mode, the ordname should contain a lowercase base32hex encoded representation of the salted & iterated hash of the full record name. **pdnssec hash-zone-record zone record** can be used to calculate this hash.

12.9 Security

During typical PowerDNSSEC operation, the private part of the signing keys are 'online', which can be compared to operating an HTTPS server, where the certificate is available on the webserver for cryptographic purposes.

In some settings, having such (private) keying material available online is considered undesirable. In this case, consider running in pre-signed mode.

12.10 Performance

As of version 3.0, PowerDNSSEC has not been tuned for performance. In practice this means that operations above 1000 queries per second should not yet be attempted.

12.11 Thanks to, acknowledgements

PowerDNSSEC has been made possible by the help & contributions of many people. We would like to thank:

- Peter Koch (DENIC)
 - Olaf Kolkman (NLNetLabs)
 - Wouter Wijngaards (NLNetLabs)
 - Marco Davids (SIDN)
 - Markus Travaille (SIDN)
 - Antoin Verschuren (SIDN)
 - Olafur Guðmundsson (IETF)
 - Dan Kaminsky (Recursion Ventures)
 - Roy Arends (Nominet)
 - Miek Gieben
 - Stephane Bortzmeyer (AFNIC)
 - Michael Braunoeder (nic.at)
 - Peter van Dijk
 - Maik Zumstrull
 - Jose Arthur Benetasso Villanova
 - Stefan Schmidt (CCC ;-))
 - Roland van Rijswijk (Surfnet)
 - Paul Bakker (Brainspark/Fox-IT)
 - Mathew Hennessy
 - Johannes Kuehrer (Austrian World4You GmbH)
 - Marc van de Geijn (bHosted.nl)
 - Stefan Arentz
-

- Martin van Hensbergen (Fox-IT)
 - Christoph Meerwald
 - Leen Besselink
 - Detlef Peeters
 - Christof Meerwald
 - Jack Lloyd
 - Frank Altpeter
 - Fredrik Danerklint
 - Vasilij G Tolstov
 - Brielle Bruns
 - Evan Hunt (ISC)
 - Ralf van der Enden
 - .. this list is far from complete yet ..
-

Chapter 13

TSIG: shared secret authorization and authentication

Note

Available since PowerDNS Authoritative Server 3.0!

TSIG, as defined in RFC 2845, is a method for signing DNS messages using shared secrets. Each TSIG shared secret has a name, and PowerDNS can be told to allow zone transfer of a domain if the request is signed with an authorized name.

In PowerDNS, TSIG shared secrets are stored by the various backends. In case of the popular Generic backends, they can be found in the 'tsigkeys' table. The name can be chosen freely, but the algorithm name will typically be 'hmac-md5'. The content is a Base64-encoded secret.

Note

Most backends require DNSSEC support enabled to support TSIG. For the Generic SQL Backend make sure to use the DNSSEC enabled schema and to turn on the relevant '-dnssec' flag (for example, gmysql-dnssec)!

13.1 Provisioning outbound AXFR access

To actually provision a named secret permission to AXFR a zone, set a metadata item in the 'domainmetadata' table called 'TSIG-ALLOW-AXFR' with the key name in the content field.

As an example:

```
sql> insert into tsigkeys (name, algorithm, secret) values ('test', 'hmac-md5', 'kp4/24 ↵
  gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=');
sql> select id from domains where name='powerdnssec.org';
5
sql> insert into domainmetadata (domain_id, kind, content) values (5, 'TSIG-ALLOW-AXFR', ' ↵
  test');

$ dig -t axfr powerdnssec.org @127.0.0.1 -y 'test:kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3 ↵
  oRSP7ys='
```

To ease interoperability, the equivalent configuration above in BIND would look like this:

```
key test. {
    algorithm hmac-md5;
    secret "kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=";
```

```
};  
  
zone "powerdnssec.org" {  
    type master;  
    file "powerdnssec.org";  
    allow-transfer { key test.; };  
};
```

A packet authorized and authenticated by a TSIG signature will gain access to a zone even if the remote IP address is not otherwise allowed to AXFR a zone.

13.2 Provisioning signed notification and AXFR requests

To configure PowerDNS to send out TSIG signed AXFR requests for a zone to its master(s), set the AXFR-MASTER-TSIG metadata item for the relevant domain to the key that must be used.

The actual TSIG key must also be provisioned, as outlined in the previous section.

For the popular Generic SQL backends, configuring the use of TSIG for AXFR requests could be achieved as follows:

```
sql> insert into tsigkeys (name, algorithm, secret) values ('test', 'hmac-md5', 'kp4/24 ↵  
    gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=');  
sql> select id from domains where name='powerdnssec.org';  
5  
sql> insert into domainmetadata (domain_id, kind, content) values (5, 'AXFR-MASTER-TSIG', ' ↵  
    test');
```

This setup corresponds to the TSIG-ALLOW-AXFR access rule defined in the previous section.

In the interest of interoperability, the configuration above is (not quite) similar to the following BIND statements:

```
key test. {  
    algorithm hmac-md5;  
    secret "kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=";  
};  
  
server 127.0.0.1 {  
    keys { test.; };  
};  
  
zone "powerdnssec.org" {  
    type slave;  
    masters { 127.0.0.1; };  
    file "powerdnssec.org";  
};
```

Except that in this case, TSIG will be used for all communications with the master, not just those about AXFR requests.

Chapter 14

AXFR ACLs

Starting with the PowerDNS Authoritative Server 3.1, per-zone AXFR ACLs can be stored in the domainmetadata table. Each ACL row can list one subnet (v4 or v6), or the magical value 'AUTO-NS' that tries to allow all potential slaves in.

Example:

```
sql> select id from domains where name='example.com';
7
sql> insert into domainmetadata (domain_id, kind, content) values (7,'ALLOW-AXFR-FROM',' ←
  AUTO-NS');
sql> insert into domainmetadata (domain_id, kind, content) values (7,'ALLOW-AXFR-FROM ←
  ','2001:db8::/48');
```

Chapter 15

Per zone settings aka Domain Metadata

Starting with the PowerDNS Authoritative Server 3.0, each served zone can have "metadata". Such metadata determines how this zone behaves in certain circumstances.

**Warning**

Domain metadata is only available for DNSSEC capable backends! Make sure to enable the proper '-dnssec' setting to benefit, and to have performed the DNSSEC schema update.

Most of these metadata items are described elsewhere in the documentation. The following settings are available:

ALLOW-AXFR-FROM Per-zone AXFR ACLs (see Chapter 14).

AXFR-MASTER-TSIG Use this named TSIG key to retrieve this zone from its master (see Section 13.2).

LUA-AXFR-SCRIPT Script to be used to edit incoming AXFRs (see Section 18.2.2).

NSEC3NARROW Determines if this zone operates in NSEC3 'narrow' mode (see 'set-nsec3' in Section 12.5).

NSEC3PARAM NSEC3 parameters of a DNSSEC zone. Will be used to synthesize the NSEC3PARAM record. If present, NSEC3 is used, if not present, zones default to NSEC (see 'set-nsec3' in Section 12.5).

PRESIGNED This zone carries DNSSEC RRSIGs (signatures), and is presigned (see 'set-presigned' in Section 12.5).

SOA-EDIT When serving this zone, modify the SOA serial number in one of several ways. Mostly useful to get slaves to re-transfer a zone regularly to get fresh RRSIGs.

Available modes are: INCEPTION (which sets the SOA Serial to the current two-week signing period start in seconds since the UNIX epoch), INCEPTION-WEEK (number of weeks since the epoch), INCREMENT-WEEKS (which increments the serial with the number of weeks since the epoch), EPOCH (number of seconds since the epoch). Finally, INCEPTION-EPOCH (available since 3.1) is special and sets the new SOA serial number to the maximum of the old SOA serial number, and age in seconds of the start of the current signing period.

TSIG-ALLOW-AXFR Allow these named TSIG keys to AXFR this zone (see Section 13.1).

Chapter 16

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.

Take care not to point **recursor** to the PowerDNS Authoritative Server itself, 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**.

16.1 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. This means that for questions for which there is no answer, PowerDNS will consult the recursor for an recursive query, even if PowerDNS is authoritative for a domain! This will only cause problems if you 'fake' domains which don't really exist.

If you want to create such fake domains or override existing domains, please set the **allow-recursion-override** feature (available as of 2.9.14).

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 mail servers 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 17

PowerDNS Recursor: a high performance resolving nameserver

The PowerDNS recursor is part of the source tarball of the main PowerDNS distribution, but it is released separately. Starting from the version 3.0 pre-releases, there are zero known bugs or issues with the recursor. It is known to power the resolving needs of over 100 million internet connections.

The documentation below is only for the 3.0 series, users of older versions are urged to upgrade!

Notable features:

- Uses MTasker ([homepage](#))
- Can handle thousands of concurrent questions. A quad Xeon 3GHz has been measured functioning very well at 40000 real life replayed packets per second, with 40% cpu idle. More testing equipment is needed to max out the recursor.
- Powered by a highly modern DNS packet parser that should be resistant against many forms of buffer overflows.
- Best spoofing protection that we know about, involving both source port randomisation and spoofing detection.
- Uses 'connected' UDP sockets which allow the recursor to react quickly to unreachable hosts or hosts for which the server is running, but the nameserver is down. This makes the recursor faster to respond in case of misconfigured domains, which are sadly very frequent.
- Special support for FreeBSD, Linux and Solaris stateful multiplexing (kqueue, epoll, completion ports, /dev/poll).
- Very fast, and contains innovative query-throttling code to save time talking to obsolete or broken nameservers.
- 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, when enabled to do so with `--verbose`.
- The algorithm is simple and quite nifty.

The PowerDNS recursor is controlled and queried using the `rec_control` tool.

17.1 pdns_recursor settings

At startup, the recursing nameserver reads the file `recursor.conf` from the configuration directory, often `/etc/powerdns` or `/usr/local/etc`. Each setting below can appear on the command line, prefixed by `--`, or in the configuration file. The command line overrides the configuration file.

A switch can be set to on simply by passing it, like `--daemon`, and turned off explicitly by `--daemon=off` or `--daemon=no`.

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-existence. Can double the amount of queries needed. Off by default.
- allow-from** Comma separated netmasks (both IPv4 and IPv6) that are allowed to use the server. The default allows access only from RFC 1918 private IP addresses, like 10.0.0.0/8. Due to the aggressive nature of the internet these days, it is highly recommended to not open up the recursor for the entire internet. Questions from IP addresses not listed here are ignored and do not get an answer.
- allow-from-file** Like **allow-from**, except reading from file. Overrides the 'allow-from' setting. To use this feature, supply one netmask per line, with optional comments preceded by a #. Available since 3.1.5.
- auth-can-lower-ttl** Authoritative zones can transmit a TTL value that is lower than that specified in the parent zone. This is called a 'delegation inconsistency'. To follow RFC 2181 paragraphs 5.2 and 5.4 to the letter, enable this feature. This will mean a slight deterioration of performance, and it will not solve any problems, but does make the recursor more standards compliant. Not recommended unless you have to tick an 'RFC 2181 compliant' box. Off by default.
- auth-zones** Comma separated list of 'zonename=filename' pairs. Zones read from these files are served authoritatively. Example: **auth-zones= ds9a.nl=/var/zones/ds9a.nl, powerdns.com=/var/zones/powerdns.com**. Available since 3.1.
- chroot** If set, chroot to this directory for more security. See Chapter 7.
Make sure that /dev/log is available from within the chroot. Logging will silently fail over time otherwise (on logrotate).
- client-tcp-timeout** Time to wait for data from TCP clients. Defaults to 2 seconds.
- config-dir** Directory where the configuration file can be found.
- daemon** Operate in the background, which is the default.
- delegation-only** A Verisign special.
- disable-packetcache** Turn off the packet cache. Useful when running with Lua scripts that can not be cached. Available since 3.2.
- dont-query** The DNS is a public database, but sometimes contains delegations to private IP addresses, like for example 127.0.0.1. This can have odd effects, depending on your network, and may even be a security risk. Therefore, since version 3.1.5, the PowerDNS recursor by default does not query private space IP addresses. This setting can be used to expand or reduce the limitations.
- entropy-source** From version 3.1.5 onwards, PowerDNS can read entropy from a (hardware) source. This is used for generating random numbers which are very hard to predict. Generally on UNIX platforms, this source will be /dev/urandom, which will always supply random numbers, even if entropy is lacking. Change to /dev/random if PowerDNS should block waiting for enough entropy to arrive.
- export-etc-hosts** If set, this flag will export the host names and IP addresses mentioned in /etc/hosts. Available since 3.1.
- export-etc-hosts-suffix** If set, all hostnames in the export-etc-hosts file are loaded in canonical form, based on this suffix, unless the name contain a '.', in which case the name is unchanged. So an entry called 'pc' with export-etc-hosts-suffix='home.com' will lead to the generation of 'pc.home.com' within the recursor. An entry called 'server1.home' will be stored as 'server1.home', regardless of the export-etc-hosts setting. Available in since 3.4.
- fork** If running on an SMP system with enough memory, this feature forks PowerDNS so it benefits from two processors. Experimental. Renames controlsockets, so care is needed to connect to the right one using **rec_control**, using **--socket-pid**. Available in versions of the Recursor before 3.2, replaced by the 'threads' setting.
- forward-zones** Comma separated list of 'zonename=IP' pairs. Queries for zones listed here will be forwarded to the IP address listed. **forward-zones= ds9a.nl=213.244.168.210, powerdns.com=127.0.0.1**. Available since 3.1.
Since version 3.1.5, multiple IP addresses can be specified. Additionally, port numbers other than 53 can be configured. Sample syntax: **forward-zones=ds9a.nl=213.244.168.210:5300;127.0.0.1, powerdns.com=127.0.0.1;9.8.7.6:530**, or on the command line: **--forward-zones="ds9a.nl=213.244.168.210:5300;127.0.0.1, powerdns.com=127.0.0.1;9.8.7.6:530"**, Forwarded queries have the 'recursion desired' bit set to 0, meaning that this setting is intended to forward queries to authoritative servers.
-

forward-zones-file Same as **forward-zones**, parsed from a file. Only 1 zone is allowed per line, specified as follows: **ds9a.nl=213.244.1.1.2.3.4:5300**. No comments are allowed. Available since 3.1.5.

Since 3.2, zones prefixed with a '+' are forwarded with the recursion-desired bit set to one, for which see 'forward-zones-recurse'. Default behaviour without '+' is as with 'forward-zones'.

forward-zones-recurse Like regular 'forward-zones' (see above), but forwarded queries have the 'recursion desired' bit set to 1, meaning that this setting is intended to forward queries to authoritative servers or to resolving servers. Available since 3.2.

hint-file If set, the root-hints are read from this file. If unset, default root hints are used. Available since 2.9.19.

local-address Local IPv4 or IPv6 addresses to bind to, comma separated. Defaults to only loopback. Addresses can also contain port numbers, for IPv4 specify like this: **1.2.3.4:5300**, for IPv6: **:::1:5300**. Port specifications are available since 3.1.2.

**Warning**

When binding to wildcard addresses, UNIX semantics mean that answers may not be sent from the address a query was received on. It is highly recommended to bind to explicit addresses.

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

log-common-errors Some DNS errors occur rather frequently and are no cause for alarm. Logging these is on by default.

logging-facility If set to a digit, logging is performed under this LOCAL facility. See Section 6.3. Available from 3.1.3 and onwards. Do not pass names like 'local0'!

max-cache-entries Maximum number of DNS cache entries. 1 million per thread will generally suffice for most installations.

max-packetcache-entries Maximum number of Packet Cache entries. 1 million per thread will generally suffice for most installations. Available since 3.2.

max-cache-ttl Maximum number of seconds to cache an item in the DNS cache, no matter what the original TTL specified. Available since 3.2.

max-negative-ttl A query for which there is authoritatively no answer is cached to quickly deny a record's existence later on, without putting a heavy load on the remote server. In practice, caches can become saturated with hundreds of thousands of hosts which are tried only once. This setting, which defaults to 3600 seconds, puts a maximum on the amount of time negative entries are cached.

max-tcp-clients Maximum number of simultaneous incoming TCP connections allowed. Defaults to 128. Available since 2.9.18.

max-tcp-per-client Maximum number of simultaneous incoming TCP connections allowed per client (remote IP address). Defaults to 0, which means unlimited.

network-timeout Number of milliseconds to wait for a remote authoritative server to respond. Defaults to 1500 msec, available since 3.2.

packetcache-ttl Maximum number of seconds to cache an item in the packet cache, no matter what the original TTL specified. Available since 3.2.

packetcache-servfail-ttl Maximum number of seconds to cache a 'server failure' answer in the packet cache. Available since 3.2.

query-local-address Send out local queries from this address, or addresses. Since 3.2, by adding multiple addresses, increased spoofing resilience is achieved. Addresses can be separated by a comma.

query-local-address6 Send out local IPv6 queries from this address or addresses Disabled by default, which also disables outgoing IPv6 support. Since version 3.2, multiple addresses can be specified, separated by a comma.

quiet Don't log queries. On by default.

- remotes-ringbuffer-entries** Number of entries in the remotes ringbuffer, which keeps statistics on who is querying your server. Can be read out using **rec_control top-remotes**. Defaults to 0.
- serve-rfc1918** On by default, this makes the server authoritatively aware of: `10.in-addr.arpa`, `168.192.in-addr.arpa`, `16-31.172.in-addr.arpa`, which saves load on the AS112 servers. Individual parts of these zones can still be loaded or forwarded.
- server-id** The PowerDNS recursor by replies to a query for 'id.server' with its hostname, useful for in clusters. Use this setting to override the answer it gives.
- setgid, setuid** PowerDNS can change its user and group id after binding to its socket. Can be used for better security.
- socket-dir** Where to store the control socket. This option also works with the controller, **rec_control**.
- socket-owner, socket-group, socket-mode** Owner, group and mode of the controlsocket. Owner and group can be specified by name, mode is in octal.
- spoofer-nearmiss-max** If set to non-zero, PowerDNS will assume it is being spoofed after seeing this many answers with the wrong id. Defaults to 20.
- trace** If turned on, output impressive heaps of logging. May destroy performance under load.
- version** Print version of this binary. Useful for checking which version of the PowerDNS recursor is installed on a system. Available since 3.1.5.
- version-string** By default, PowerDNS replies to the 'version.bind' query with its version number. Security conscious users may wish to override the reply PowerDNS issues.

17.2 pdns_recursor command line

All configuration settings from the previous section can also be passed on the command line, and will override the configuration file. In addition, the following options make sense on the command line:

- config** Emit a default configuration file.
- help** Output all configuration settings and command line flags.

17.3 Controlling and querying the recursor

To control and query the PowerDNS recursor, the tool `rec_control` is provided. This program talks to the recursor over the 'controlsocket', often stored in `/var/run`.

As a sample command, try:

```
# rec_control ping
pong
```

When not running as root, **--socket-dir=/tmp** might be appropriate.

All `rec_control` commands are documented below:

- dump-cache filename** Dumps the entire cache to the filename mentioned. This file should not exist already, PowerDNS will refuse to overwrite it. While dumping, the recursor will not answer questions.
- get statistic** Retrieve a statistic. For items that can be queried, see below.
- get-all** Retrieve all statistics in one go. Available since 3.2.
- get-parameter parameter1 parameter2 ..** Retrieve a configuration parameter. All parameters from the configuration and command line can be queried. Available since 3.2.

ping Check if server is alive.

quit Request shutdown of the recursor.

reload-zones Reload data about all authoritative and forward zones. The configuration file is also scanned to see if the **auth-domain**, **forward-domain** and **export-etc-hosts** statements have changed, and if so, these changes are incorporated.

top-remotes Shows the top-20 most active remote hosts. Statistics are over the last 'remotes-ringbuffer-entries' queries, which defaults to 0.

wipe-cache domain0. [domain1. domain2.] Wipe entries from the cache. This is useful if, for example, an important server has a new IP address, but the TTL has not yet expired. Multiple domain names can be passed. For versions before 3.1, you must terminate a domain with a **!**. So to wipe powerdns.org, issue 'rec_control wipe-cache powerdns.org.'. For later versions, the dot is optional.

Note that deletion is exact, wiping 'com.' will leave 'www.powerdns.com.' untouched!

**Warning**

As of 3.1.7, this command also wipes the negative query cache for the specified domain.

**Warning**

Don't just wipe 'www.somedomain.com', its NS records or CNAME target may still be undesired, so wipe 'somedomain.com' as well.

The command 'get' can query a large number of statistics, which are detailed in Section [17.6](#).

More details on what 'throttled' queries and the like are can be found below in Section [17.5](#).

17.4 PowerDNS Recursor performance

To get the best out of the PowerDNS recursor, which is important if you are doing thousands of queries per second, please consider the following.

- Limit the size of the caches to a sensible value. Cache hit rate does not improve meaningfully beyond 4 million **max-cache-entries** per thread, reducing the memory footprint reduces CPU cache misses. See below for more information about the various caches.
- Compile using g++ 4.1 or later. This compiler really does a good job on PowerDNS, much better than 3.4 or 4.0.
- On AMD/Intel hardware, wherever possible, run a 64-bit binary. This delivers a nearly twofold performance increase. On UltraSPARC, there is no need to run with 64 bits.
- Consider performing a 'profiled build' as described in the README. This is good for a 20% performance boost in some cases.
- When running with >3000 queries per second, and running Linux versions prior to 2.6.17 on some motherboards, your computer may spend an inordinate amount of time working around an ACPI bug for each call to gettimeofday. This is solved by rebooting with 'clock=tsc' or upgrading to a 2.6.17 kernel.

The above is relevant if dmesg shows **Using pmtmr for high-res timesource**

- A busy server may need hundreds of file descriptors on startup, and deals with spikes better if it has that many available later on. Linux by default restricts processes to 1024 file descriptors, which should suffice most of the time, but Solaris has a default limit of 256. This can be raised using the ulimit command. FreeBSD has a default limit that is high enough for even very heavy duty use.
-

- For older versions <3.2: If you need it, try **--fork**, this will fork the daemon into two halves, allowing it to benefit from a second CPU. This feature almost doubles performance, but is a bit of a hack.
- for 3.2 and higher, set 'threads' to your number of CPUs.
- For best PowerDNS Recursor performance, use a recent version of your operating system, since this generally offers the best event multiplexer implementation available (kqueue, epoll, ports or /dev/poll).
- A Recursor under high load puts a severe stress on any stateful (connection tracking) firewall, so much so that the firewall may fail.

Specifically, many Linux distributions run with a connection tracking firewall configured. For high load operation (thousands of queries/second), It is advised to either turn off iptables completely, or use the 'NOTRACK' feature to make sure DNS traffic bypasses the connection tracking.

Sample Linux command lines would be:

```
# iptables -t raw -I OUTPUT -p udp --dport 53 -j NOTRACK
# iptables -t raw -I PREROUTING -p udp --sport 53 -j NOTRACK
# iptables -I INPUT -p udp --dport 53 -j ACCEPT
# iptables -I INPUT -p udp --sport 53 -j ACCEPT
# iptables -I OUTPUT -p udp --dport 53 -j ACCEPT

# # optionally
# ip6tables -t raw -I OUTPUT -p udp --dport 53 -j NOTRACK
# ip6tables -t raw -I PREROUTING -p udp --sport 53 -j NOTRACK
# ip6tables -I INPUT -p udp --dport 53 -j ACCEPT
# ip6tables -I INPUT -p udp --sport 53 -j ACCEPT
# ip6tables -I OUTPUT -p udp --dport 53 -j ACCEPT
```

Following the instructions above, you should be able to attain very high query rates.

17.4.1 Recursor Caches

The PowerDNS Recursor contains a number of caches, or information stores:

Nameserver speeds cache The "NSSpeeds" cache contains the average latency to all remote authoritative servers.

Negative cache The "Negcache" contains all domains known not to exist, or record types not to exist for a domain.

Recursor Cache The Recursor Cache contains all DNS knowledge gathered over time.

Packet Cache The Packet Cache contains previous answers sent to clients. If a question comes in that matches a previous answer, this is sent back directly.

The Packet Cache is consulted first, immediately after receiving a packet. This means that a high hitrate for the Packet Cache automatically lowers the cache hitrate of subsequent caches. This explains why releases 3.2 and beyond see dramatically lower DNS cache hitrates, since this is the first version with a Packet Cache.

17.5 Details

17.5.1 Anti-spoofing

The PowerDNS recursor 3.0 uses a fresh UDP source port for each outgoing query, making spoofing around 64000 times harder. This raises the bar from 'easily doable given some time' to 'very hard'. Under some circumstances, 'some time' has been measured at 2 seconds. This technique was first used by `dnscache` by Dan J. Bernstein.

In addition, PowerDNS detects when it is being sent too many unexpected answers, and mistrusts a proper answer if found within a clutch of unexpected ones.

This behaviour can be tuned using the **spoofer-nearmiss-max**.

17.5.2 Throttling

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.

17.6 Statistics

The `rec_control get` command can be used to query the following statistics, either single keys or multiple statistics at once:

<code>all-outqueries</code>	counts the number of outgoing UDP queries since starting
<code>answers0-1</code>	counts the number of queries answered within 1 millisecond
<code>answers100-1000</code>	counts the number of queries answered within 1 second
<code>answers10-100</code>	counts the number of queries answered within 100 milliseconds
<code>answers1-10</code>	counts the number of queries answered within 10 milliseconds
<code>answers-slow</code>	counts the number of queries answered after 1 second
<code>cache-bytes</code>	Size of the cache in bytes (since 3.3.1)
<code>cache-entries</code>	shows the number of entries in the cache
<code>cache-hits</code>	counts the number of cache hits since starting
<code>cache-misses</code>	counts the number of cache misses since starting
<code>chain-resends</code>	number of queries chained to existing outstanding query
<code>client-parse-errors</code>	counts number of client packets that could not be parsed
<code>concurrent-queries</code>	shows the number of MThreads currently running
<code>dlg-only-drops</code>	number of records dropped because of delegation only setting
<code>dont-outqueries</code>	number of outgoing queries dropped because of 'dont-query' setting (← since 3.3)
<code>ipv6-outqueries</code>	number of outgoing queries over IPv6
<code>max-mthread-stack</code>	maximum amount of thread stack ever used
<code>negcache-entries</code>	shows the number of entries in the Negative answer cache
<code>noerror-answers</code>	counts the number of times it answered NOERROR since starting
<code>nsspeeds-entries</code>	shows the number of entries in the NS speeds map
<code>nsset-invalidations</code>	number of times an nsset was dropped because it no longer worked
<code>nxdomain-answers</code>	counts the number of times it answered NXDOMAIN since starting
<code>outgoing-timeouts</code>	counts the number of timeouts on outgoing UDP queries since starting
<code>over-capacity-drops</code>	Questions dropped because over maximum concurrent query limit (since ← 3.2)
<code>packetcache-bytes</code>	Size of the packet cache in bytes (since 3.3.1)
<code>packetcache-entries</code>	Size of packet cache (since 3.2)
<code>packetcache-hits</code>	Packet cache hits (since 3.2)
<code>packetcache-misses</code>	Packet cache misses (since 3.2)
<code>qa-latency</code>	shows the current latency average, in microseconds
<code>questions</code>	counts all End-user initiated queries with the RD bit set
<code>resource-limits</code>	counts number of queries that could not be performed because of ← resource limits
<code>server-parse-errors</code>	counts number of server replied packets that could not be parsed
<code>servfail-answers</code>	counts the number of times it answered SERVFAIL since starting
<code>spooft-prevents</code>	number of times PowerDNS considered itself spoofed, and dropped the ← data
<code>sys-msec</code>	number of CPU milliseconds spent in 'system' mode
<code>tcp-client-overflow</code>	number of times an IP address was denied TCP access because it already ← had too many connections

tcp-outqueries	counts the number of outgoing TCP queries since starting
tcp-questions	counts all incoming TCP queries (since starting)
throttled-out	counts the number of throttled outgoing UDP queries since starting
throttle-entries	shows the number of entries in the throttle map
unauthorized-tcp	number of TCP questions denied because of allow-from restrictions
unauthorized-udp	number of UDP questions denied because of allow-from restrictions
unexpected-packets	number of answers from remote servers that were unexpected (might point ← to spoofing)
uptime	number of seconds process has been running (since 3.1.5)
user-msec	number of CPU milliseconds spent in 'user' mode

In the `rrd/` subdirectory a number of `rrdtool` scripts is provided to make nice graphs of all these numbers. Use **rec_control get-all** to get all statistics in one go.

It should be noted that `answers0-1 + answers1-10 + answers10-100 + answers100-1000 + packetcache-hits + over-capacity-drops = questions`.

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% ←
cache hits, outpacket/query ratio 37%, 12% throttled
```

This means that there are 13971 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.

17.7 Scripting

As of version 3.1.7 of the PowerDNS Recursor, it is possible to modify resolving behaviour using simple scripts written in the [Lua](#) programming language.



Warning

This functionality is expected to change from version to version as additional scripting needs become apparent!

These scripts can be used to quickly override dangerous domains, for load balancing or for legal or commercial purposes.

As of 3.1.7, queries can be intercepted in two places: before the resolving logic starts to work, plus after the resolving process failed to find a correct answer for a domain.

17.7.1 Configuring Lua scripts

In order to load scripts, the PowerDNS Recursor must have Lua support built in. The packages distributed from the PowerDNS website have this language enabled, other distributions may differ. To compile with Lua support, use: `LUA=1 make` or `LUA=1 gmake` as the case may be. Paths to the Lua include files and binaries may be found near the top of the `Makefile`.

If Lua support is available, a script can be configured either via the configuration file, or at runtime via the **rec_control** tool. Scripts can be reloaded or unloaded at runtime with no interruption in operations. If a new script contains syntax errors, the old script remains in force.

On the command line, or in the configuration file, the setting **lua-dns-script** can be used to supply a full path to a 'lua' script.

At runtime, **rec_control reload-lua-script** can be used to either reload the script from its current location, or, when passed a new file name, load one from a new location. A failure to parse the new script will leave the old script in working order.

Finally, **rec_control unload-lua-script** can be used to remove the currently installed script, and revert to unmodified behaviour.

17.7.2 Writing Lua PowerDNS Recursor scripts

Once a script is loaded, PowerDNS looks for several functions, as detailed below. All of these functions are optional.

`preresolve (remoteip, domain, qtype)` is called before any DNS resolution is attempted, and if this function indicates it, it can supply a direct answer to the DNS query, overriding the internet. This is useful to combat botnets, or to disable domains unacceptable to an organization for whatever reason.

`postresolve (remoteip, domain, qtype, records, origrcode)` is called right before returning a response to a client (and, unless `setvariable()` is called, to the packet cache too). It allows inspection and modification of almost any detail in the return packet. Available since 3.4.

`function nxdomain (remoteip, domain, qtype)` is called after the DNS resolution process has run its course, but ended in an 'NXDOMAIN' situation, indicating that the domain or the specific record does not exist. This can be used for various purposes.

`function nodata (remoteip, domain, qtype, records)` is just like `nxdomain`, except it gets called when a domain exists, but the requested type does not. This is where one would implement DNS64. Available since 3.4.

All these functions are passed the IP address of the requester, plus the name and type being requested. In return, these functions indicate if they have taken over the request, or want to let normal proceedings take their course.



Warning

In development versions of the PowerDNS Recursor, versions which were never released except as for testing purposes, these functions had a fourth parameter: `localip`. This parameter has been replaced by `getlocaladdress()`, for which see below.

If a function has taken over a request, it should return an `rcode` (usually 0), and specify a table with records to be put in the answer section of a packet. An interesting `rcode` is `NXDOMAIN` (3, or `pdns.NXDOMAIN`), which specifies the non-existence of a domain. Returning -1 and an empty table signifies that the function chose not to intervene.

A minimal sample script:

```
function nxdomain ( ip, domain, qtype )
  print ("nxhandler called for: ", ip, domain, qtype)

  ret={}
  if qtype ~= pdns.A then return -1, ret end -- only A records
  if not string.find(domain, "^www%.") then return -1, ret end -- only things that start ←
    with www.
  if not matchnetmask(ip, "10.0.0.0/8", "192.168.0.0/16") then return -1, ret end -- only ←
    interfere with local queries
  ret[1]={qtype=pdns.A, content="127.1.2.3"} -- add IN A 127.1.2.3
  ret[2]={qtype=pdns.A, content="127.3.2.1"} -- add IN A 127.3.2.1
  setvariable()
  return 0, ret -- return no error, plus records
end
```



Warning

Please do NOT use the above sample script in production! Responsible NXDomain redirection requires more attention to detail.

Note that the domain is passed to the Lua function terminated by a `'.'`. A more complete sample script is provided as `powerdns-examp.lua` in the PowerDNS Recursor distribution.

The answer content format is (nearly) identical to the storage in the PowerDNS Authoritative Server database, or as in zone files. The exception is that, unlike in the database, there is no `'prio'` field, which means that an MX record with priority 25 pointing to `'smtp.mailserver.com'` would be encoded as `'25 smtp.mailserver.com.'`

Useful return `'rcodes'` include 0 for "no error" and `pdns.NXDOMAIN` for "NXDOMAIN".

Fields that can be set in the return table include:

content Content of the record, as specified above in `'zone file format'`. No default, mandatory field.

place Place of this record. Defaults to 1, indicating `'Answer'` section. Can also be 2, for Authority or 3 for Additional. When using this rare feature, always emit records with `'Place'` in ascending order. This field is usually not needed.

qname qname of the answer, the `'name'` of the record. Defaults to the name of the query, which is almost always correct except when specifying additional records or rolling out a CNAME chain.

qtype Currently the numerical qtype of the answer, defaulting to `'1'` which is an A record. Can be also be specified as `pdns.A`, or `pdns.CNAME` etc.

ttd Time to live of a record. Defaults to 3600. Be sure not to specify differing TTLs within answers with an identical qname. While this will be encoded in DNS, actual results may be undesired.



Warning

The result table must have indexes that start at 1! Otherwise the first or confusingly the last entry of the table will be ignored. A useful technique is to return data using: `return 0, {{qtype=1, content="1.2.3.4"}, {qtype=1, content="4.3.2.1"}}` as this will get the numbering right automatically.

The function `matchnetmask(ip, netmask1, netmask2..)` (or `matchnetmask(ip, {netmask1, netmask2})`) is available to match incoming queries against a number of netmasks. If any of these match, the function returns true.

To log messages with the main PowerDNS Recursor process, use `pdnslog(message)`. Available since 3.2.

To retrieve the IP address on which a query was received, use `getlocaladdress()`. Available since 3.2.

To indicate that an answer should not be cached in the packet cache, use `setvariable()`. Available since 3.3.

To get fake AAAA records for DNS64 usage, use `return "getFakeAAAARecords", domain, "fe80::21b:77f-f:0:0"`. Available since 3.4.

17.8 Design and Engineering of the PowerDNS Recursor



Warning

This section is aimed at programmers wanting to contribute to the recursor, or to help fix bugs. It is not required reading for a PowerDNS operator, although it might prove interesting.

The PowerDNS Recursor consists of very little code, the core DNS logic is less than a thousand lines.

This smallness is achieved through the use of some fine infrastructure: `MTasker`, `MOADNSParser`, `MPlexer` and the C++ Standard Library/Boost. This page will explain the conceptual relation between these components, and the route of a packet through the program.

17.8.1 The PowerDNS Recursor

The Recursor started out as a tiny project, mostly a technology demonstration. These days it consists of the core plus 9000 lines of features. This combined with a need for very high performance has made the recursor code less accessible than it was. The page you are reading hopes to rectify this situation.

17.8.2 Synchronous code using MTasker

The original name of the program was **syncres**, which is still reflected in the file name `syncres.cc`, and the class `SyncRes`. This means that PowerDNS is written naively, with one thread of execution per query, synchronously waiting for packets. Normally this would lead to very bad performance (unless running on a computer with very fast threading, like possibly the Sun CoolThreads family), so PowerDNS employs **MTasker** for very fast userspace threading.

MTasker, which was developed separately from PowerDNS, does not provide a full multithreading system but restricts itself to those features a nameserver needs. It offers cooperative multitasking, which means there is no forced preemption of threads. This in turn means that no two **MThreads** ever really run at the same time.

This is both good and bad, but mostly good. It means PowerDNS does not have to think about locking. No two threads will ever be talking to the DNS cache at the same time, for example.

It also means that the recursor could block if any operation takes too long.

The core interaction with MTasker are the `waitEvent()` and `sendEvent()` functions. These pass around `PacketID` objects. Everything PowerDNS needs to wait for is described by a `PacketID` event, so the name is a bit misleading. Waiting for a TCP socket to have data available is also passed via a `PacketID`, for example.

The version of MTasker in PowerDNS is newer than that described at the MTasker site, with a vital difference being that the `waitEvent()` structure passes along a copy of the exact `PacketID` `sendEvent()` transmitted. Furthermore, threads can trawl through the list of events being waited for and modify the respective `PacketIDs`. This is used for example with **near miss** packets: packets that appear to answer questions we asked, but differ in the DNS id. On seeing such a packet, the recursor trawls through all `PacketIDs` and if it finds any nearmisses, it updates the `PacketID::nearMisses` counter. The actual `PacketID` thus lives inside MTasker while any thread is waiting for it.

17.8.3 MPlexer

The Recursor uses a separate socket per outgoing query. This has the important benefit of making spoofing 64000 times harder, and additionally means that ICMP errors are reported back to the program. In measurements this appears to happen to one in ten queries, which would otherwise take a two-second timeout before PowerDNS moves on to another nameserver.

However, this means that the program routinely needs to wait on hundreds or even thousands of sockets. Different operating systems offer various ways to monitor the state of sockets or more generally, file descriptors. To abstract out the differing strategies (`select`, `epoll`, `kqueue`, `completion ports`), PowerDNS contains **MPlexer** classes, all of which descend from the `FDMultiplexer` class.

This class is very simple and offers only five important methods: `addReadFD()`, `addWriteFD()`, `removeReadFD()`, `removeWriteFD()` and `run`.

The arguments to the **add** functions consist of an `fd`, a callback, and a `boost::any` variable that is passed as a reference to the callback.

This might remind you of the MTasker above, and it is indeed the same trick: state is stored within the MPlexer. As long as a file descriptor remains within either the Read or Write active list, its state will remain stored.

On arrival of a packet (or more generally, when an FD becomes readable or writable, which for example might mean a new TCP connection), the callback is called with the aforementioned reference to its parameter.

The callback is free to call `removeReadFD()` or `removeWriteFD()` to remove itself from the active list.

PowerDNS defines such callbacks as `newUDPQuestion()`, `newTCPConnection()`, `handleRunningTCPConnection()`.

Finally, the `run()` method needs to be called whenever the program is ready for new data. This happens in the main loop in `pdns_recursor.cc`. This loop is what MTasker refers to as **the kernel**. In this loop, any packets or other MPlexer events get translated either into new MThreads within MTasker, or into calls to `sendEvent()`, which in turn wakes up other MThreads.

17.8.4 MOADNSParser

Yes, this does stand for **the Mother of All DNS Parsers**. And even that name does not do it justice! The MOADNSParser is the third attempt I've made at writing DNS packet parser and after two miserable failures, I think I've finally gotten it right.

Writing and parsing DNS packets, and the DNS records it contains, consists of four things:

1. Parsing a DNS record (from packet) into memory
2. Generating a DNS record from memory (to packet)
3. Writing out memory to user-readable zone format
4. Reading said zone format into memory

This gets tedious very quickly, as one needs to implement all four operations for each new record type, and there are dozens of them.

While writing the MOADNSParser, it was discovered there is a remarkable symmetry between these four transitions. DNS Records are nearly always laid out in the same order in memory as in their zone format representation. And reading is nothing but inverse writing.

So, the MOADNSParser is built around the notion of a **Conversion**, and we write all Conversion types once. So we have a Conversion from IP address in memory to an IP address in a DNS packet, and vice versa. And we have a Conversion from an IP address in zone format to memory, and vice versa.

This in turn means that the entire implementation of the ARecordContent is as follows (wait for it!)

```
conv.xfrIP(d_ip);
```

Through the use of the magic called `c++ Templates`, this one line does everything needed to perform the four operations mentioned above.

At one point, I got really obsessed with PowerDNS memory use. So, how do we store DNS data in the PowerDNS recursor? I mentioned **memory** above a lot - this means we could just store the DNSRecordContent objects. However, this would be wasteful.

For example, storing the following:

```
www.ds9a.nl 3600 IN CNAME outpost.ds9a.nl.
```

Would duplicate a lot of data. So, what is actually stored is a partial DNS packet. To store the CNAMEDNSRecordContent that corresponds to the above, we generate a DNS packet that has **www.ds9a.nl IN CNAME** as its question. Then we add **3600 IN CNAME outpost.ds9a.nl.** as its answer. Then we chop off the question part, and store the rest in the **www.ds9a.nl IN CNAME** key in our cache.

When we need to retrieve **www.ds9a.nl IN CNAME**, the inverse happens. We find the proper partial packet, prefix it with a question for **www.ds9a.nl IN CNAME**, and expand the resulting packet into the answer **3600 IN CNAME outpost.ds9a.nl.**

Why do we go through all these motions? Because of DNS compression, which allows us to omit the whole **.ds9a.nl.** part, saving us 9 bytes. This is amplified when storing multiple MX records which all look more or less alike. This optimization is not performed yet though.

Even without compression, it makes sense as all records are automatically stored very compactly.

The PowerDNS recursor only parses a number of **well known record types** and passes all other information across verbatim - it doesn't have to know about the content it is serving.

17.8.5 The C++ Standard Library / Boost

C++ is a powerful language. Perhaps a bit too powerful at times, you can turn a program into a real freakshow if you so desire. PowerDNS generally tries not to go overboard in this respect, but we do build upon a very advanced part of the Boost C++ library: `boost::multi index container`.

This container provides the equivalent of SQL indexes on multiple keys. It also implements compound keys, which PowerDNS uses as well.

The main DNS cache is implemented as a multi index container object, with a compound key on the name and type of a record. Furthermore, the cache is sequenced, each time a record is accessed it is moved to the end of the list. When cleanup is performed, we start at the beginning. New records also get inserted at the end. For DNS correctness, the sort order of the cache is case insensitive.

The multi index container appears in other parts of PowerDNS, and MTasker as well.

17.8.6 Actual DNS Algorithm

The DNS RFCs do define the DNS algorithm, but you can't actually implement it exactly that way, it was written in 1987.

Also, like what happened to HTML, it is expected that even non-standards conforming domains work, and a sizable fraction of them is misconfigured these days.

Everything begins with `SyncRes::beginResolve()`, which knows nothing about sockets, and needs to be passed a domain name, dns type and dns class which we are interested in. It returns a vector of `DNSResourceRecord` objects, ready for writing either into an answer packet, or for internal use.

After checking if the query is for any of the hardcoded domains (`localhost`, `version.bind`, `id.server`), the query is passed to `SyncRes::doResolve`, together with two vital parameters: the `depth` and `beenthere` set. As the word **recursor** implies, we will need to recurse for answers. The **depth** parameter documents how deep we've recursed already.

The `beenthere` set prevents loops. At each step, when a nameserver is queried, it is added to the `beenthere` set. No nameserver in the set will ever be queried again for the same question in the recursion process - we know for a fact it won't help us further. This prevents the process from getting stuck in loops.

`SyncRes::doResolve` first checks if there is a CNAME in cache, using `SyncRes::doCNAMECacheCheck`, for the domain name and type queried and if so, changes the query (which is passed by reference) to the domain the CNAME points to. This is the cause of many DNS problems, a CNAME record really means **start over with this query**.

This is followed by a call `do SyncRes::doCacheCheck`, which consults the cache for a straight answer to the question (as possibly rerouted by a CNAME). This function also consults the so called negative cache, but we won't go into that just yet.

If this function finds the correct answer, and the answer hasn't expired yet, it gets returned and we are (almost) done. This happens in 80 to 90% of all queries. Which is good, as what follows is a lot of work.

To recap:

1. `beginResolve()` - entry point, does checks for hardcoded domains
2. `doResolve()` - start of recursion process, gets passed `depth` of 0 and empty `beenthere` set
3. `doCNAMECacheCheck()` - check if there is a CNAME in cache which would reroute the query
4. `doCacheCheck()` - see if cache contains straight answer to possibly rerouted query.

If the data we were queried for was in the cache, we are almost done. One final step, which might as well be optional as nobody benefits from it, is `SyncRes::addCruft`. This function does additional processing, which means that if the query was for the MX record of a domain, we also add the IP address of the mail exchanger.

17.8.7 The non-cached case

This is where things get interesting, because we start out with a nearly empty cache and have to go out to the net to get answers to fill it.

The way DNS works, if you don't know the answer to a question, you find somebody who does. Initially you have no other place to go than the root servers. This is embodied in the `SyncRes::getBestNSNamesFromCache` method, which gets passed the domain we are interested in, as well as the `depth` and `beenthere` parameters mentioned earlier.

From now on, assume our query will be for `www.powerdns.com.`. `SyncRes::getBestNSNamesFromCache` will first check if there are NS records in cache for `www.powerdns.com.`, but there won't be. It then checks `powerdns.com.` NS, and while these records do exist on the internet, the recursor doesn't know about them yet. So, we go on to check the cache for `com.` NS, for which the same holds. Finally we end up checking for `.` NS, and these we do know about: they are the root servers and were loaded into PowerDNS on startup.

So, `SyncRes::getBestNSNamesFromCache` fills out a set with the **names** of nameservers it knows about for the `.` zone.

This set, together with the original query `www.powerdns.com` gets passed to `SyncRes::doResolveAt`. This function can't yet go to work immediately though, it only knows the names of nameservers it can try. This is like asking for directions and instead of hearing **take the third right** you are told **go to 123 Fifth Avenue, and take a right** - the answer doesn't help you further unless you know where 123 Fifth Avenue is.

`SyncRes::doResolveAt` first shuffles the nameservers both randomly and on performance order. If it knows a nameserver was fast in the past, it will get queried first. More about this later.

Ok, here is the part where things get a bit scary. How does `SyncRes::doResolveAt` find the IP address of a nameserver? Well, by calling `SyncRes::getAs` (**get A records**), which in turn calls.. `SyncRes::doResolve`. Hang on! That's where we came from! Massive potential for loops here. Well, it turns out that for any domain which can be resolved, this loop terminates. We do pass the `beenthere` set again, which makes sure we don't keep on asking the same questions to the same nameservers.

Ok, `SyncRes::getAs` will give us the IP addresses of the chosen root-server, because these IP addresses were loaded on startup. We then ask these IP addresses (nameservers can have several) for its best answer for `www.powerdns.com.`. This is done using the `LWRes` class and specifically `LWRes::asyncreolve`, which gets passed domain name, type and IP address. This function interacts with `MTasker` and `Mplexer` above in ways which needn't concern us now. When it returns, the `LWRes` object contains the best answers the queried server had for our domain, which in this case means it tells us about the nameservers of `com.`, and their IP addresses.

All the relevant answers it gives are stored in the cache (or actually, merged), after which `SyncRes::doResolveAt` (which we are still in) evaluates what to do now.

There are 6 options:

1. The final answer is in, we are done, return to `SyncRes::doResolve` and `SyncRes::beginResolve`
2. The nameserver we queried tells us the domain we asked for authoritatively does not exist. In case of the root-servers, this happens when we query for `www.powerdns.kom.` for example, there is no `kom.`. Return to `SyncRes::beginResolve`, we are done.
3. A lesser form - it tells us it is authoritative for the query we asked about, but there is no record matching our type. This happens when querying for the IPv6 address of a host which only has an IPv4 address. Return to `SyncRes::beginResolve`, we are done.
4. The nameserver passed us a CNAME to another domain, and we need to reroute. Go to `SyncRes::doResolve` for the new domain.
5. The nameserver did not know about the domain, but does know who does, a *referral*. Stay within `doResolveAt` and loop to these new nameservers.
6. The nameserver replied saying *no idea*. This is called a *lame delegation*. Stay within `SyncRes::doResolveAt` and try the other nameservers we have for this domain.

When not redirected using a CNAME, this function will loop until it has exhausted all nameservers and all their IP addresses. DNS is surprisingly resilient that there is often only a single non-broken nameserver left to answer queries, and we need to be prepared for that.

This is the whole DNS algorithm in PowerDNS, all in less than 700 lines of code. It contains a lot of tricky bits though, related to the cache.

17.8.8 Some of the things we glossed over

Whenever a packet is sent to a remote nameserver, the response time is stored in the `SyncRes::s_nsSpeeds` map, using an exponentially weighted moving average. This EWMA averages out different response times, and also makes them decrease over time. This means that a nameserver that hasn't been queried recently gradually becomes **faster** in the eyes of PowerDNS, giving it a chance again.

A timeout is accounted as a 1s response time, which should take that server out of the running for a while.

Furthermore, queries are throttled. This means that each query to a nameserver that has failed is accounted in the `s_throttle` object. Before performing a new query, the query and the nameserver are looked up via `shouldThrottle`. If so, the query is assumed to have failed without even being performed. This saves a lot of network traffic and makes PowerDNS quick to respond to lame servers.

It also offers a modicum of protection against birthday attack powered spoofing attempts, as PowerDNS will not inundate a broken server with queries.

The negative query cache we mentioned earlier caches the cases 2 and 3 in the enumeration above. This data needs to be stored separately, as it represents **non-data**. Each negcache query entry is the name of the SOA record that was presented with the evidence of non-existence. This SOA record is then retrieved from the regular cache, but with the TTL that originally came with the NXDOMAIN (case 2) or NXRRSET (case 3).

17.8.9 The Recursor Cache

As mentioned before, the cache stores partial packets. It also stores not the **Time To Live** of records, but in fact the **Time To Die**. If the cache contains data, but it is expired, that data should not be deemed present. This bit of PowerDNS has proven tricky, leading to deadlocks in the past.

There are some other very tricky things to deal with. For example, through a process called **more details**, a domain might have more nameservers than listed in its parent zone. So, there might only be two nameservers for `powerdns.com.` in the `com.` zone, but the `powerdns.com` zone might list more.

This means that the cache should not, when talking to the `com.` servers later on, overwrite these four nameservers with only the two copies the `com.` servers pass us.

However, in other cases (like for example for SOA and CNAME records), new data should overwrite old data.

Note that PowerDNS deviates from RFC 2181 (section 5.4.1) in this respect.

17.8.10 Some small things

The server-side part of PowerDNS (`pdns_recursor.cc`), which listens to queries by end-users, is fully IPv6 capable using the `ComboAddress` class. This class is in fact a union of a `struct sockaddr_in` and a `struct sockaddr_in6`. As long as the `sin_family` (or `sin6_family`) and `sin_port` members are in the same place, this works just fine, allowing us to pass a `ComboAddress*`, cast to a `sockaddr*` to the socket functions. For convenience, the `ComboAddress` also offers a `length()` method which can be used to indicate the length - either `sizeof(sockaddr_in)` or `sizeof(sockaddr_in6)`.

Access to the recursor is governed through the `NetmaskGroup` class, which internally contains `Netmask`, which in turn contain a `ComboAddress`.

Chapter 18

Master/Slave operation & replication

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

18.1 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.

18.2 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 unrefresh 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, retransferring 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.

Since 2.9.21, PowerDNS supports multiple masters. For the BIND backend, the native BIND configuration language suffices to specify multiple masters, for SQL based backends, list all master servers separated by commas in the 'master' field of the domains table.

18.2.1 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.

Before a supermaster notification succeeds, the following conditions must be met:

- The supermaster must carry a SOA record for the notified domain
- The supermaster IP must be present in the 'supermaster' table
- The set of NS records for the domain, as retrieved by the slave from the supermaster, must include the name that goes with the IP address in the supermaster table

So, to benefit from 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 the account name out but it does help keep track of where a domain comes from.

18.2.2 Modifying a slave zone using a script

As of version 3.0, the PowerDNS Authoritative Server can invoke a Lua script on an incoming AXFR zone transfer. The user-defined function `axfrfilter` within your script is invoked for each resource record read during the transfer, and the outcome of the function defines what PowerDNS does with the records.

(idea and documentation contributed by Jan-Piet Mens)

What you can accomplish using a Lua script:

- Ensure consistent values on SOA
- Change incoming SOA serial number to a YYYYMMDDnn format
- Ensure consistent NS RRset
- Timestamp the zone transfer with a TXT record

To enable a Lua script for a particular slave zone, determine the `domain_id` for the zone from the `domains` table, and add a row to the `domainmetadata` table for the domain. Supposing the domain we want has an `id` of 3, the following SQL statement will enable the Lua script `my.lua` for that domain:

```
INSERT INTO domainmetadata (domain_id, kind, content) VALUES (3, "LUA-AXFR-SCRIPT", "/lua ←  
/my.lua");
```

The Lua script must both exist and be syntactically correct; if not, the zone transfer is not performed.

Your Lua functions have access to the query codes through a pre-defined Lua table called `pdns`. For example if you want to check for a CNAME record you can either compare `qtype` to the numeric constant 5 or the value `pdns.CNAME` -- they are equivalent.

If your function decides to handle a resource record it must return a result code of 0 together with a Lua table containing one or more replacement records to be stored in the back-end database. If, on the other hand, your function decides not to modify a record, it must return -1 and an empty table indicating that PowerDNS should handle the incoming record as normal.

Consider the following simple example:

```

function axfrfilter(remoteip, zone, qname, qtype, ttl, prio, content)

-- Replace each HINFO records with this TXT
if qtype == pdns.HINFO then
  resp = {}
  resp[1] = {   qname   = qname,
               qtype   = pdns.TXT,
               ttl     = 99,
               content  = "Hello Ahu!"
             }
  return 0, resp
end

-- Grab each _tstamp TXT record and add a time stamp
if qtype == pdns.TXT and string.starts(qname, "_tstamp.") then
  resp = {}
  resp[1] = {
    qname   = qname,
    qtype   = qtype,
    ttl     = ttl,
    content  = os.date("Ver %Y%m%d-%H:%M")
  }
  return 0, resp
end

resp = {}
return -1, resp
end

function string.starts(s, start)
  return s.sub(s, 1, s.len(start)) == start
end

```

Upon an incoming AXFR, PowerDNS calls our `axfrfilter` function for each record. All HINFO records are replaced by a TXT record with a TTL of 99 seconds and the specified string. TXT Records with names starting with `_tstamp.` get their value (`_rdata_`) set to the current time stamp. All other records are unhandled.

18.3 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 19

Fancy records for seamless email and URL integration

**Warning**

As of PowerDNS Authoritative Server 3.0, fancy records are no longer supported!

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 20

Index of all Authoritative Server settings

All PDNS Authoritative Server 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**.

allow-recursion-override=on|off By specifying **allow-recursion-override**, local data even about hosts that don't exist will override the internet. This allows you to generate zones that don't really exist on the internet. Does increase the number of SQL queries for hosts that truly don't exist, also not in your database.

cache-ttl=... Seconds to store packets in the PacketCache. See Section [9.3.1](#).

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](#).

do-ipv6-additional-processing=... Perform AAAA additional processing.

fancy-records=... Process URL and MBOXFW records. See Chapter [19](#).

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 [16](#).

-
- 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-ipv6=...** Local IPv6 address to which we bind. You can specify multiple addresses separated by commas or whitespace.
- 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 digit, logging is performed under this LOCAL facility. See Section [6.3](#). Available from 1.99.9 and onwards. Do not pass names like 'local0'!
- loglevel=...** Amount of logging. Higher is more. Do not set below 3
- master [=on].** Turn on master support. Boolean.
- max-cache-entries** Maximum number of cache entries. 1 million will generally suffice for most installations. Available since 2.9.22.
- max-queue-length=...** If this many packets are waiting for database attention, consider the situation hopeless and respawn.
- max-tcp-connections=...** Allow this many incoming TCP DNS connections simultaneously.
- 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 [9.3.2](#).
- no-config** Do not attempt to read the configuration file.
- no-shuffle** Do not attempt to shuffle query results.
- server-id** This is the server ID that will be returned on an EDNS NSID query. Defaults to the host name.
- out-of-zone-additional-processing | --out-of-zone-additional-processing=yes | --out-of-zone-additional-processing=no** Do out of zone additional processing. This means that if a malicious user adds a '.com' zone to your server, it is not used for other domains and will not contaminate answers. Do not enable this setting if you run a public DNS service with untrusted users. Off by default.
- query-cache-ttl=...** Seconds to store queries with an answer in the Query Cache. See Section [9.3.2](#).
- 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-local-address6=...** Source IP address for sending IPv6 queries.
- 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.
- queue-limit=...** Maximum number of milliseconds to queue a query. See Chapter [9](#).
- recursive-cache-ttl=...** Seconds to store recursive packets in the PacketCache. See Section [9.3.1](#).
- recursor=...** If set, recursive queries will be handed to the recursor specified here. See Chapter [16](#).
- send-root-referral | --send-root-referral=yes | --send-root-referral=no | --send-root-referral=lean** If set, PowerDNS will send out old-fashioned root-referrals when queried for domains for which it is not authoritative. Wastes some bandwidth but may solve incoming query floods if domains are delegated to you for which you are not authoritative, but which are queried by broken recursors. Available since 2.9.19.
Since 2.9.21, it is possible to specify 'lean' root referrals, which waste less bandwidth.
-

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.

slave-cycle-interval=60 Schedule slave up-to-date checks of domains whose status is unknown every .. seconds.

smtpredirector=... Our smtpredir MX host. See Chapter 19.

soa-expire-default=604800 Default SOA expire.

soa-minimum-ttl=3600 Default SOA minimum ttl.

soa-refresh-default=10800 Default SOA refresh.

soa-retry-default=3600 Default SOA retry.

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-conforming AXFRs, which are slow, but may be necessary to placate some old client tools.

urlredirector=... Where we send hosts to that need to be url redirected. See Chapter 19.

version-string=anonymous|powerdns|full|custom When queried for its version over DNS (**dig chaos txt version.bind @pdns.ip.addr**) PowerDNS normally responds truthfully. With this setting you can overrule what will be returned. Set the **version-string** to 'full' to get the default behaviour, to 'powerdns' to just make it state 'served by PowerDNS - http://www.powerdns.com'. The 'anonymous' setting will return a ServFail, much like Microsoft nameservers do. You can set this response to a custom value as well.

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 21

Index of all Authoritative Server metrics

21.1 Counters & variables

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

21.1.1 Counters

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

21.1.2 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 webservice and are in fact not even collected without the webservice 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 22

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.



Warning

Host names and the MNAME of a SOA records are NEVER terminated with a '.' in PowerDNS storage! If a trailing '.' is present it will inevitably cause problems, problems that may be hard to debug.

The PowerDNS Recursor can serve and store all record types, regardless of whether these are explicitly supported.

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'.

AFSDB (since 2.9.21) Specialised record type for the 'Andrew Filesystem'. Stored as: '#subtype hostname', where subtype is a number.

CERT (since 2.9.21) Specialised record type for storing certificates, defined in RFC 2538.

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'.

DNSKEY (since 2.9.21) The DNSKEY DNSSEC record type is fully supported, as described in RFC 3757. Note that while PowerDNS can store, retrieve and serve DNSSEC records, no further DNSSEC processing is performed.

DS (since 2.9.21) The DS DNSSEC record type is fully supported, as described in RFC 3757. Note that while PowerDNS can store, retrieve and serve DNSSEC records, no further DNSSEC processing is performed.

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

KEY (since 2.9.21) The KEY record is fully supported. For its syntax, see RFC 2535.

LOC The LOC record is fully supported. For its syntax, see RFC 1876. A sample content would be: '51 56 0.123 N 5 54 0.000 E 4.00m 1.00m 10000.00m 10.00m'

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.
- NSEC (since 2.9.21)** The NSEC DNSSEC record type is fully supported, as described in RFC 3757. Note that while PowerDNS can store, retrieve and serve DNSSEC records, no further DNSSEC processing is performed.
- 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.
- RRSIG (since 2.9.21)** The RRSIG DNSSEC record type is fully supported, as described in RFC 3757. Note that while PowerDNS can store, retrieve and serve DNSSEC records, no further DNSSEC processing is performed.
- 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:

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)

Table 22.1: SOA fields

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. Make sure to check whether your backend of choice supports Autoserail.

- SPF (since 2.9.21)** SPF records can be used to store Sender Policy Framework details (RFC 4408).
- SSHFP (since 2.9.21)** The SSHFP record type, used for storing Secure Shell (SSH) fingerprints, is fully supported. A sample from RFC 4255 is: '2 1 123456789abcdef67890123456789abcdef67890'.
- 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 priority field and '100 389 mars.conaxis.ch' in the content field.
- TXT** The TXT field can be used to attach textual data to a domain. Text is stored plainly.

Chapter 23

HOWTO & Frequently Asked Questions

This chapter contains a number of FAQs and HOWTOs.

23.1 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).

The PowerDNS company provides free support on the public mailing lists, and can help or support you in private as well. For first class and rapid support, please contact powerdns-support@netherlabs.nl, or see www.powerdns.com

More information about the PowerDNS community, and its mailing lists, can be found on [its Wiki](#). On the wiki, you will also find information on how to file bugs.

Below, please find a list of common questions asked on our public mailing lists.

Q: Help! A: Please try harder :-). Specifically, before people will be able to help you, they need to know a lot about your system. If you list more details, chances are you'll get better answers.

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 problem. 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.

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 mailing list. Please email the authors directly only if you previously entered a support contract with them, or are considering doing so. For mailing list details, see [the mailing lists page](#).

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.

Q: My information is confidential, must I send it to the mailing list? If you desire privacy, please consider entering a support relationship with us, in which case we invite you to contact powerdns.support.sales@netherlabs.eu.

23.2 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!

If you don't see your question answered here, please check out [the Wiki FAQ](#), but do note that it is user-editable and not under our constant control.

Q: I get this entry a lot of times in my log file: Authoritative empty NO ERROR to 1.2.3.4 for 'powerdns.nl' (AAAA).. As the name implies, this is not an error. It tells you there are questions for a domain which exists in your database, but for which no record of the requested type exists. To get rid of this error, add `log-dns-details=off` to your configuration.

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,mysql'`.

Q: PowerDNS does not give authoritative answers, how come? A: This is almost always not the case. An authoritative answer is recognized by the 'AA' bit being set. Many tools prominently print the number of Authority records included in an answer, leading users to conclude that the absence or presence of these records indicates the authority of an answer. This is not the case.

Verily, many misguided country code domain operators have fallen into this trap and demand authority records, even though these are fluff and quite often misleading. Invite such operators to look at section 6.2.1 of RFC 1034, which shows a correct authoritative answer without authority records. In fact, none of the non-deprecated authoritative answers shown have authority records!

Sorry for sounding like DJB on this, but we get so many misguided questions about authority..

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.

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'm trying to build from SVN 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. More information for developers is available on [the PowerDNS Open Source Community Wiki](#).

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.

Q: PowerDNS crashes when I install the pdns-static .deb on Debian SID A: Indeed. Install the .deb files 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: I see this a lot: Backend error: Failed to execute mysql_query, perhaps connection died? A: Check your MySQL timeout, it may be set too low. This can be changed in the `my.cnf` file.

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 20 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.

23.3 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.

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 occurred 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 choosing.

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: 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 is this 'AhuException' I keep reading about? A: This name has historical reasons and has **no significance**.

Q: I need a backend but I can't write it, can you help? A: Yes, we also do custom development. Contact us at powerdns.support@sales or visit www.powerdns.com

23.4 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: 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](#)) mailing list 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](#)) mailing list. 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 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.

Chapter 24

Other tools included with PowerDNS

PowerDNS comes with several tools that can be used to do various DNS related things.

24.1 Notification proxy (nproxy)

Available in PowerDNS 2.9.22 and later.

For additional security, operators may prefer to have a 'hidden slave' that sits behind a strong firewall. This slave pulls in zones from the outside world, and stores them in a database. This database is then used by publicly accessible nameservers to publish zone data.

For proper slave operation, master nameservers send out notifications to inform slaves of updates. This is not normally a problem, but when operating with a hidden slave behind a firewall, notification packets can't reach the slave.

For this purpose, the PowerDNS also supplies a notification proxy. It sits outside the firewall, and accepts notifications from remote master servers. It interprets and validates these packets, and then sends on a new notification to the hidden slave.

The hidden slave then promptly retrieves an updated zone from the master.

The notification proxy, called **nproxy**, can be configured using the following settings:

chroot Change root to this directory for additional security.

daemon Run in the background. Defaults to true, can be turned off using '--daemon=no'.

listen-address Public addresses (IPv4 and IPv6) to listen on for incoming notification packets. Defaults to "all addresses", but it is highly recommended to specify addresses here.

origin-address Can be used to pin the address the nproxy uses to communicate with the hidden slave. Highly recommended. Corresponds to the PowerDNS setting **trusted-notification-proxy**.

powerdns-address IP address (IPv4 or IPv6) of the hidden slave, to which notifications should be relayed. This setting is mandatory, and has no default.

setuid and setgid Change to these numerical user-id and/or group-id, dropping root privileges, for additional security.

Chapter 25

Tools to analyse DNS traffic

DNS is highly mission critical, it is therefore necessary to be able to study and compare DNS traffic. Since 2.9.18, PowerDNS comes with three tools to aid in analysis:

**Warning**

As of 2.9.18 these tools are somewhat rough - they have no help messages for example. They do work though.

dnsreplay pcapfile [ipaddress] [port number] This program takes recorded questions and answers and replays them to a specified nameserver and reporting afterwards which percentage of answers matched, were worse or better.

dnswasher pcapfile output Anonymises recorded traffic, making sure it only contains DNS, and that the originating IP addresses of queries are stripped, which may allow you to send traces to our company or mailing list without violating obligations towards your customers or privacy laws.

dnsscope pcapfile Calculates statistics without replaying traffic

Appendix A

Backends in detail

This appendix lists several of the available backends in more detail

A.1 PipeBackend

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	No
Case	Depends
DNSSEC	Partial, no delegation, no key storage
Module name	pipe
Launch name	pipe

Table A.1: PipeBackend capabilities

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

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 deploy 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)\$'. Please be aware that the single quotes used in this document should not be present in the configuration file, and only on the command line. In the configuration file, the previous example would be stored as: pipe-regex=^www.powerdns.com;(A|ANY)\$

Available since 2.8.

pipebackend-abi-version This is the version of the question format that is sent to the co-process (pipe-command) for the pipe backend.

If not set the default pipebackend-abi-version is 1. When set to 2, the local-ip-address field is added after the remote-ip-address. (the local-ip-address refers to the IP address the question was received on). When set to 3, the real remote IP/subnet is added based on edns-subnet support (this also requires enabling 'edns-subnet-processing').

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. Questions and answers are terminated by single newline ('\n') characters.

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. For abi-version 2 or 3, PowerDNS sends 'HELO\t2' or 'HELO\t3'. 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 type:

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, for type Q questions:

pipebackend-abi-version = 1 [default]

```
Q qname  qclass  qtype id  remote-ip-address
```

pipebackend-abi-version = 2

```
Q qname  qclass  qtype id  remote-ip-address local-ip-address
```

pipebackend-abi-version = 3

```
Q qname  qclass  qtype id  remote-ip-address local-ip-address edns-subnet-address
```

Fields are tab separated, and terminated with a single \n. The remote-ip-address is the IP address of the nameserver asking the question; the local-ip-address is the IP address on which the question was received.

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 unless you want to support AXFR.

remote-ip-address is the ip-address of the nameserver asking the question. local-ip-address is the ip-address that was queried locally. edns-subnet-address is the actual client subnet as provided via edns-subnet support. Note that for the SOA query that precedes an AXFR, edns-subnet is always set to 0.0.0.0/0.

AXFR-queries look like this:

```
AXFR id
```

The id is gathered from the answer to a SOA query.

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 of sending 'END' without anything else. The answer format (for abi-version 1 and 2):

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

'content' is as specified in Chapter 22. For MX and SRV, content consists of the priority, followed by a tab, followed by the actual content.

A sample dialogue may look like this (note that in reality, almost all queries will actually be for the ANY qtype):

```
Q www.ds9a.nl IN CNAME -1 213.244.168.210
DATA www.ds9a.nl IN CNAME 3600 1 ws1.ds9a.nl
END
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.

For abi-version 3, DATA-responses get two extra fields:

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

scopebits indicates how many bits from the subnet provided in the question (originally from edns-subnet) were used in determining this answer. This can aid caching (although PowerDNS does not currently use this value). The auth field indicates whether this response is authoritative; this is for DNSSEC.

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($line eq "HELO\t1") {
    print "FAIL\n";
    print STDERR "Received '$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 ($stype,$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.1.2 Notes

Besides regular query types, the DNS also knows the 'ANY' query type. When a server receives a question for this ANY type, it should reply with all record types available.

Backends should therefore implement being able to answer 'ANY' queries in this way, and supply all record types they have when they receive such an 'ANY' query. This is reflected in the sample script above, which for every qtype answers if the type matches, or if the query is for 'ANY'.

However, since backends need to implement the ANY query anyhow, PowerDNS makes use of this. Since almost all DNS queries internally need to be translated first into a CNAME query and then into the actual query, possibly followed by a SOA or NS query (this is how DNS works internally), it makes sense for PowerDNS to speed this up, and just ask the ANY query of a backend.

When it has done so, it gets the data about SOA, CNAME and NS records in one go. This speeds things up tremendously.

The upshot of the above is that for any backend, including the PIPE backend, implementing the ANY query is NOT optional. And in fact, a backend may see almost exclusively ANY queries. This is not a bug.

A.2 Random Backend

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	No
Case	Depends
DNSSEC	Yes, no key storage
Module name	built in
Launch name	random

Table A.2: Random Backend capabilities

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.3 Generic MySQL and PgSQL backends

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.



Warning

Host names and the MNAME of a SOA records are NEVER terminated with a '.' in PowerDNS storage! If a trailing '.' is present it will inevitably cause problems, problems that may be hard to debug.

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 % 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.

Native	Yes
Master	Yes
Slave	Yes
Superslave	Yes
Autoserial	Yes (v3.1 and up)
Case	All lower
DNSSEC	Yes (set <code>gmysql-dnssec</code> or <code>gpgsql-dnssec</code>)
Module name < 2.9.3	<code>pgmysql</code>
Module name > 2.9.2	<code>gmysql</code> and <code>gpgsql</code>
Launch name	<code>gmysql</code> and <code>gpgsql2</code> and <code>gpgsql</code>

Table A.3: Generic PostgreSQL and MySQL backend capabilities

A.3.1 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(128) 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)
) Engine=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(10) DEFAULT NULL,
  content     VARCHAR(64000) DEFAULT NULL,
  ttl         INT DEFAULT NULL,
  prio        INT DEFAULT NULL,
  change_date INT DEFAULT NULL,
  primary key (id)
) Engine=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
```

```
) Engine=InnoDB;
```

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

To support or migrate to DNSSEC, the following SQL statements must be executed:

```
create table domainmetadata (
  id      INT auto_increment,
  domain_id      INT NOT NULL,
  kind      VARCHAR(16),
  content     TEXT,
  primary key(id)
);

create index domainmetaidindex on domainmetadata(domain_id);

create table cryptokeys (
  id      INT auto_increment,
  domain_id      INT NOT NULL,
  flags     INT NOT NULL,
  active     BOOL,
  content     TEXT,
  primary key(id)
);

create index domainidindex on cryptokeys(domain_id);

alter table records add ordername      VARCHAR(255);
alter table records add auth bool;
create index orderindex on records(ordername);

create table tsigkeys (
  id      INT auto_increment,
  name     VARCHAR(255),
  algorithm VARCHAR(50),
  secret   VARCHAR(255),
  primary key(id)
);

create unique index namealgoindex on tsigkeys(name, algorithm);
alter table records change column type type VARCHAR(10);
```

For full migration notes, please see [Section 12.3](#).

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.

When using the InnoDB storage engine, we suggest adding the following lines to the 'create table records' command above:

```
CONSTRAINT `records_ibfk_1` FOREIGN KEY (`domain_id`) REFERENCES `domains`
(`id`) ON DELETE CASCADE
```

This automates deletion of records on deletion of a domain from the domains table.

A.3.2 PostgreSQL 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(128) 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(10) DEFAULT NULL,
    content     VARCHAR(65535) 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,
    CONSTRAINT c_lowercase_name CHECK (((name)::text = lower((name)::text)))
);

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;

```

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

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.

To support DNSSEC or to migrate to DNSSEC, the following statements have to be issued:

```

alter table records add ordername VARCHAR(255);
alter table records add auth bool;
create index orderindex on records(ordername);

create table domainmetadata (
    id SERIAL PRIMARY KEY,
    domain_id INT REFERENCES domains(id) ON DELETE CASCADE,
    kind VARCHAR(16),
    content TEXT
);

create index domainidmetaindex on domainmetadata(domain_id);

create table cryptokeys (

```

```

id SERIAL PRIMARY KEY,
domain_id INT REFERENCES domains(id) ON DELETE CASCADE,
flags INT NOT NULL,
active BOOL,
content TEXT
);
create index domainidindex on cryptokeys(domain_id);

-- GRANT ALL ON domainmetadata TO pdns;
-- GRANT ALL ON domainmetadata_id_seq TO pdns;
-- GRANT ALL ON cryptokeys TO pdns;
-- GRANT ALL ON cryptokeys_id_seq TO pdns;

create table tsigkeys (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255),
  algorithm VARCHAR(50),
  secret VARCHAR(255)
);

create unique index namealgorithindex on tsigkeys(name, algorithm);

-- GRANT ALL ON tsigkeys TO pdns;
-- GRANT ALL ON tsigkeys_id_seq TO pdns;
alter table records alter column type type VARCHAR(10);

```

For full migration notes, please see [Section 12.3](#).

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.3.3 Oracle specifics

Generic Oracle support is only available since version 2.9.18. The default setup conforms to the following schema, which you should add to an Oracle database. You may need or want to add 'namespace' statements.

```

create table domains (
  id NUMBER,
  name VARCHAR(255) NOT NULL,
  master VARCHAR(128) 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)
);
create sequence DOMAINS_ID_SEQUENCE;
create index DOMAINS$NAME on Domains (NAME);

CREATE TABLE records (
  id number(11) not NULL,
  domain_id INT DEFAULT NULL REFERENCES Domains(ID) ON DELETE CASCADE,
  name VARCHAR(255) DEFAULT NULL,
  type VARCHAR(10) DEFAULT NULL,
  content VARCHAR2(4000) DEFAULT NULL,
  ttl INT DEFAULT NULL,
  prio INT DEFAULT NULL,
  change_date INT DEFAULT NULL,

```

```

primary key (id)
);

create index RECORDS$NAME on RECORDS (NAME);
create sequence RECORDS_ID_SEQUENCE;

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

```

This schema contains all elements needed for master, slave and superslave operation. Depending on which features will be used, '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.

Inserting records is a bit different compared to MySQL and PostgreSQL, you should use:

```

insert into domains (id,name,type) values (domains_id_sequence.nextval,'netherlabs.nl',' ←
NATIVE');

```

Furthermore, use the **oracle-tnsname** setting to specify which TNSNAME the Generic Oracle Backend should be connecting to. There are no **oracle-dbname**, **oracle-host** or **oracle-port** settings, their equivalent is in `/etc/tnsnames.ora`.

A.3.4 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.

ttd 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,ttd,prio,type,domain_id,name from records where type='%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,ttd,prio,type,domain_id,name from records where type='%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,ttd,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.3.5 DNSSEC queries

If DNSSEC is enabled (through the -dnssec flag on a gsql backend), many queries are replaced by slightly extended variants that also query the auth column. The auth column is always added as the rightmost column. These are the -auth defaults:

basic-query-auth Basic query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where type='%s' and name='%s'**

id-query-auth Basic with ID query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where type='%s' and name='%s' and domain_id=%d**

wildcard-query-auth Wildcard query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where type='%s' and name like '%s'**

wildcard-id-query-auth Wildcard with ID query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where type='%s' and name like '%s' and domain_id=%d'**

any-query-auth Any query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where name='%s'**

any-id-query-auth Any with ID query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where name='%s' and domain_id=%d**

wildcard-any-query-auth Wildcard ANY query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where name like '%s'**

wildcard-any-id-query-auth Wildcard ANY with ID query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where name like '%s' and domain_id=%d'**

list-query-auth AXFR query. Default: **select content,ttl,prio,type,domain_id,name, auth from records where domain_id=%d' order by name, type**

Additionally, there are some new queries to determine NSEC(3) order:

get-order-first-query DNSSEC Ordering Query, first. Default: **select ordername, name from records where domain_id=%d and ordername is not null order by 1 asc limit 1**

get-order-before-query DNSSEC Ordering Query, before. Default: **select ordername, name from records where ordername <= '%s' and domain_id=%d and ordername is not null order by 1 desc limit 1**

get-order-after-query DNSSEC Ordering Query, after. Default: **select min(ordername) from records where ordername > '%s' and domain_id=%d and ordername is not null**

get-order-last-query DNSSEC Ordering Query, last. Default: **select ordername, name from records where ordername != '' and domain_id=%d and ordername is not null order by 1 desc limit 1**

Finally, these two queries are used to set ordername and auth correctly in a database:

set-order-and-auth-query DNSSEC set ordering query. Default: **update records set ordername='%s',auth=%d where name='%s' and domain_id=%d'**

nullify-ordername-and-auth-query DNSSEC nullify ordername query. Default: **update records set ordername=NULL,auth=0 where name='%s' and type='%s' and domain_id=%d'**

Make sure to read Section [12.8.5](#) if you wish to calculate ordername and auth without using pdns-rectify.

A.3.6 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,ty 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,a 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 last_check=%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,noti 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.3.7 Fancy records



Warning

Fancy records are unsupported as of version 3.0

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 type='%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 type='%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.3.8 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 ↵
    type='%s' and name='%s'
```

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 type='%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, where one should substitute 'gmysql', 'gpgsql', 'godbc' or 'goracle' for the prefix 'backend'. So 'backend-database' can stand for 'gpgsql-database' or 'gmysql-database' etc.

backend-database Database name to connect to

backend-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.

backend-port Database port to connect to.

gmysql-socket (only for MySQL!) File name where the MySQL connection socket resides. Often `/tmp/mysql.sock` or `/var/run/mysqld/mysqld.sock`.

backend-password Password to connect with

backend-user PostgreSQL user to connect as

A.3.9 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.3.10 Slave operation

These backends are 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.3.11 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 18.2.1 for details.

A.3.12 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.4 Oracle backend

Native	Yes
Master	Yes
Slave	Yes
Superslave	Yes
Autoserial	Yes
DNSSEC	Yes
Module name	oracle
Launch name	oracle

Table A.4: Oracle backend capabilities

This is the Oracle Database backend, completely rewritten for the 3.0 release, with easily configurable SQL statements, allowing you to graft PowerDNS functionality onto any Oracle database of your choosing.

The Oracle backend is difficult, and possibly illegal, to distribute in binary form. To use it, you will probably need to compile PowerDNS from source. OCI headers are expected in \$ORACLE_HOME/rdbms/public, and OCI libraries in \$ORACLE_HOME/lib. That is where they should be with a working installation of the full Oracle Database client. Oracle InstantClient should work as well, but you will need to make the libraries and headers available in appropriate paths.

This backend uses two kinds of database connections. First, it opens a session pool. Connections from this pool are used only for queries reading DNS data from the database. Second, it opens normal (non-pooled) connections on demand for any kind of write access. The reason for this split is to allow redundancy by replication. Each DNS frontend server can have a local read-only replicated instance of your database. Open the session pool to the local replicated copy, and all data will be available with high performance, even if the main database goes down. The writing connections should go directly to the main database.

Of course, if you do not require this kind of redundancy, or want to avoid the substantial Oracle Database licensing costs, all connections can just go to the same database with the same credentials. Also, the write connections should be entirely unnecessary if you do not plan to use either master or slave mode.

The following configuration settings are available:

oracle-pool-database, oracle-pool-username, oracle-pool-password The database to use for read access. OracleBackend will try to create a session pool, so make sure this database user has the necessary permissions. If your connection requires environment variables to be set, e.g. `ORACLE_HOME`, `NLS_LANG`, or `LD_LIBRARY_PATH`, make sure these are set when PowerDNS runs. `/etc/default/pdns` might help.

oracle-master-database, oracle-master-username, oracle-master-password The database to use for write access. These are normal connections, not a session pool. The backend may open more than one at a time.

oracle-session-min, oracle-session-max, oracle-session-inc Parameters for the connection pool underlying the session pool. OCI will open `session-min` connections at startup, and open more connections as needed, `session-inc` at a time, until `session-max` connections are open.

oracle-nameserver-name This can be set to an arbitrary string that will be made available in the optional bind variable `:ns-name` for all SQL statements. You can use this to run multiple PowerDNS instances off the same database, while serving different zones.

There are many more options that are used to define the different SQL statements. These will be discussed after the reference database schema has been explained.

A.4.1 The Database Schema

You can find an example database schema in `schema.sql` in the PowerDNS source distribution. It is intended more as a starting point to come up with a schema that works well for your organisation, than as something you should run as it is. As long as the semantics of the SQL statements still work out, you can store your DNS data any way you like.

You should read this while having `schema.sql` to hand. Columns will not be specifically explained where their meaning is obvious.

Note

All FQDNs should be specified in lower case and without a trailing dot. Where things are lexicographically compared or sorted, make sure a sane ordering is used. `NLS_LANG=AMERICAN_AMERICA.AL32UTF8` should generally work well enough; when in doubt, enforce a plain ordering with `NLSSORT(value, 'NLS_SORT = BINARY')`.

A.4.1.1 Zones Table

This table lists the zones for which PowerDNS is supposed to be an authoritative nameserver, plus a small amount of information related to master/slave mode.

name The FQDN of the zone apex, e.g. `example.com`.

type Describes how PowerDNS should host the zone. Valid values are `NATIVE`, `MASTER`, and `SLAVE`. PowerDNS acts as an authoritative nameserver for the zone in all modes. In slave mode, it will additionally attempt to acquire the zone's content from a master server. In master mode, it will additionally send `NOTIFY` packets to other nameservers for the zone when its content changes.

Tip

There is a global setting to make PowerDNS send `NOTIFY` packets in slave mode.

last_check This value, updated by PowerDNS, is the unix timestamp of the last successful attempt to check this zone for freshness on the master.

refresh The number of seconds PowerDNS should wait after a successful freshness check before performing another one. This value is also found in the zone's SOA record. You may want to make sure to put the same thing in both places.

serial The serial of the version of the zone's content we are hosting now. This value is also found in the zone's SOA record. You may want to make sure to put the same thing in both places.

notified_serial The latest serial for which we have sent NOTIFY packets. Updated by PowerDNS.

A.4.1.2 The Zonemasters and ZoneAlsoNotify Tables

These are lists of hosts PowerDNS will interact with for a zone in master/slave mode. `Zonemasters` lists the hosts PowerDNS will attempt to pull zone transfers from, and accept NOTIFY packets from. `ZoneAlsoNotify` lists hosts PowerDNS will send NOTIFY packets to, in addition to any hosts that have NS records.

Host entries can be IPv4 or IPv6 addresses, in string representation. If you need to specify a port, use `1.2.3.4:5300` notation for IPv4 and brackets for IPv6: `[abcd]:1234:5300`.

A.4.1.3 The Supermasters Table

In superslave mode, PowerDNS can accept NOTIFY packets for zones that have not been defined in the zone table yet. PowerDNS will then create an entry for the zone and attempt a zone transfer. This table defines the list of acceptable sources for supernotifications.

name An identifying string for this entry. Only used for logging.

ip The alleged originating IP address of the notification.

nameserver The FQDN of an authoritative nameserver.

A supernotification will be accepted if an entry is found such that the notification came from `ip` and `nameserver` appears in an NS record for that zone.

A.4.1.4 The ZoneMetadata Table

This is a per-zone key-value store for various things PowerDNS needs to know that are not part of the zone's content or handled by other tables. Depending on your needs, you may not want this to exist as an actual table, but simulate this in PL/SQL instead.

The currently defined metadata types are:

PRESIGNED If set to 1, PowerDNS should assume that DNSSEC signatures for this zone exist in the database and use them instead of signing records itself. For a slave zone, this will also signal to the master that we want DNSSEC records when attempting a zone transfer.

NSEC3PARAM The NSEC3 hashing parameters for the zone.

TSIG-ALLOW-AXFR The value is the name of a TSIG key. A client will be allowed to AXFR from us if the request is signed with that key.

AXFR-MASTER-TSIG The value is the name of a TSIG key. Outgoing NOTIFY packets for this zone will be signed with that key.

A.4.1.5 The Tables for Cryptographic Keys

We have two of them: `TSIGKeys` for symmetric TSIG keys, and `ZoneDNSKeys` for DNSSEC signing keys.

A.4.1.6 The Records Table

The actual DNS zone contents are stored here.

zone_id The zone this records belongs to. Normally, this is obvious. When you are dealing with zone delegations, you have to insert some records into the parent zone of their actual zone. See also `auth`.

fqdn The owner name of this record. Again, this is lower case and without a trailing dot.

revfqdn This should be a string that consists of the labels of the owner name, in reverse order, with spaces instead of dots separating them, for example:

```
'www.example.com' => 'com example www'
```

This is used as a quick and dirty way to get canonical zone ordering. You can chose a more correct and much more complicated implementation instead if you prefer. In the reference schema, this is automatically set by a trigger.

fqdnhash The NSEC3 hash of the owner name. The reference schema provides code and a trigger to calculate this, but they are not production quality. The recommendation is to load the `dnsjava` classes into your database and use their facilities for dealing with DNS names and NSEC3 hashes.

ttl The TTL for the record set. This should be the same for all members of a record set, but PowerDNS will quietly use the minimum if it encounters different values.

type The type of the record, as a canonical identification string, e.g. `AAAA` or `MX`. You can set this and `content` `NULL` to indicate a name that exists, but doesn't carry any record (a so called empty non-terminal) for NSEC/NSEC3 ordering purposes.

content The data part of the DNS record, in canonical string representation, except that if this includes FQDNs, they should be specified without a trailing dot.

last_change The unix timestamp of the last change to this record. Used only for the deprecated autoserial feature. You can omit this unless you want to use that feature.

auth 0 or 1 depending on whether this record is an authoritative member of the zone specified in `zone_id`. These are the rules for determining that: A record is an authoritative member of the zone its owner name belongs to, except for DS records, which are authoritative members of the parent zone. Delegation records, that is, NS records and related A/AAAA glue records, are additionally non-authoritative members of the parent zone.

PowerDNS has a function to automatically set this. OracleBackend doesn't support that. Do it in the database.

A.4.2 The SQL Statements

A.4.2.1 Fetching DNS records

There are five queries to do this. They all share the same set of return columns:

fqdn The owner name of the record.

ttl The TTL of the record set.

type The type of the record.

content The content of the record.

zone_id The numerical identifier of the zone the record belongs to. A record can belong to two zones (delegations/glue), in which case it may be returned twice.

last_change The unix timestamp of the last time this record was changed. Can safely be set as a constant 0, unless you use the autoserial feature.

auth 1 or 0 depending on the zone membership (authoritative or not).

Record sets (records for the same name of the same type) must appear consecutively, which means **ORDER BY** clauses are needed in some places. Empty non-terminals should be suppressed.

The queries differ in which columns are restricted by **WHERE** clauses:

oracle-basic-query Looking for records based on owner name and type. Default:

```
SELECT fqdn, ttl, type, content, zone_id, last_change, auth
FROM Records
WHERE type = :type AND fqdn = lower(:name)
```

oracle-basic-id-query Looking for records from one zone based on owner name and type. Default:

```
SELECT fqdn, ttl, type, content, zone_id, last_change, auth
FROM Records
WHERE type = :type AND fqdn = lower(:name) AND zone_id = :zoneid
```

oracle-any-query Looking for records based on owner name. Default:

```
SELECT fqdn, ttl, type, content, zone_id, last_change, auth
FROM Records
WHERE fqdn = lower(:name)
  AND type IS NOT NULL
ORDER BY type
```

oracle-any-id-query Looking for records from one zone based on owner name. Default:

```
SELECT fqdn, ttl, type, content, zone_id, last_change, auth
FROM Records
WHERE fqdn = lower(:name)
  AND zone_id = :zoneid
  AND type IS NOT NULL
ORDER BY type
```

oracle-list-query Looking for all records from one zone. Default:

```
SELECT fqdn, ttl, type, content, zone_id, last_change, auth
FROM Records
WHERE zone_id = :zoneid
  AND type IS NOT NULL
ORDER BY fqdn, type
```

A.4.2.2 Zone Metadata and TSIG

oracle-get-zone-metadata-query Fetch the content of the metadata entries of type `:kind` for the zone called `:name`, in their original order. Default:

```
SELECT md.meta_content
FROM Zones z JOIN ZoneMetadata md ON z.id = md.zone_id
WHERE z.name = lower(:name) AND md.meta_type = :kind
ORDER BY md.meta_ind
```

oracle-del-zone-metadata-query Delete all metadata entries of type `:kind` for the zone called `:name`. You can skip this if you do not plan to manage zones with the **pdnssec** tool. Default:

```
DELETE FROM ZoneMetadata md
WHERE zone_id = (SELECT id FROM Zones z WHERE z.name = lower(:name))
AND md.meta_type = :kind
```

oracle-set-zone-metadata-query Create a metadata entry. You can skip this if you do not plan to manage zones with the **pdnssec** tool. Default:

```
INSERT INTO ZoneMetadata (zone_id, meta_type, meta_ind, meta_content)
VALUES (
  (SELECT id FROM Zones WHERE name = lower(:name)),
  :kind, :i, :content
)
```

oracle-get-tsig-key-query Retrieved the TSIG key specified by `:name`. Default:

```
SELECT algorithm, secret
FROM TSIGKeys
WHERE name = :name
```

A.4.2.3 DNSSEC

oracle-get-zone-keys-query Retrieve the DNSSEC signing keys for a zone. Default:

```
SELECT k.id, k.flags, k.active, k.keydata
FROM ZonedNSKeys k JOIN Zones z ON z.id = k.zone_id
WHERE z.name = lower(:name)
```

oracle-del-zone-key-query Delete a DNSSEC signing key. You can skip this if you do not plan to manage zones with the **pdnssec** tool. Default:

```
DELETE FROM ZonedNSKeys WHERE id = :keyid
```

oracle-add-zone-key-query Add a DNSSEC signing key. You can skip this if you do not plan to manage zones with the **pdnssec** tool. Default:

```
INSERT INTO ZonedNSKeys (id, zone_id, flags, active, keydata) "
VALUES (
  zonednskeys_id_seq.NEXTVAL,
  (SELECT id FROM Zones WHERE name = lower(:name)),
  :flags,
  :active,
  :content
) RETURNING id INTO :keyid
```

oracle-set-zone-key-state-query Enable or disable a DNSSEC signing key. You can skip this if you do not plan to manage zones with the **pdnssec** tool. Default:

```
UPDATE ZonedNSKeys SET active = :active WHERE id = :keyid
```

oracle-prev-next-name-query Determine the predecessor and successor of an owner name, in canonical zone ordering. See the reference implementation for the quick and dirty way, and the RFCs for the full definition of canonical zone ordering.

This statement is a PL/SQL block that writes into two of the bind variables, not a query.

Default:

```
BEGIN
  get_canonical_prev_next(:zoneid, :name, :prev, :next);
END;
```

oracle-prev-next-hash-query Given an NSEC3 hash, this call needs to return its predecessor and successor in NSEC3 zone ordering into `:prev` and `:next`, and the FQDN of the predecessor into `:unhashed`. Default:

```
BEGIN
  get_hashed_prev_next(:zoneid, :hash, :unhashed, :prev, :next);
END;
```

A.4.2.4 Incoming AXFR

oracle-zone-info-query Get some basic information about the named zone before doing master/slave things. Default:

```
SELECT id, name, type, last_check, serial, notified_serial
FROM Zones
WHERE name = lower(:name)
```

oracle-delete-zone-query Delete all records for a zone in preparation for an incoming zone transfer. This happens inside a transaction, so if the transfer fails, the old zone content will still be there. Default:

```
DELETE FROM Records WHERE zone_id = :zoneid
```

oracle-insert-record-query Insert a record into the zone during an incoming zone transfer. This happens inside the same transaction as delete-zone, so we will not end up with a partially transferred zone. Default:

```
INSERT INTO Records (id, fqdn, zone_id, ttl, type, content)
VALUES (records_id_seq.NEXTVAL, lower(:name), :zoneid, :ttl, :type, :content)
```

oracle-finalize-axfr-query A block of PL/SQL to be executed after a zone transfer has successfully completed, but before committing the transaction. A good place to locate empty non-terminals, set the `auth` bit and NSEC3 hashes, and generally do any post-processing your schema requires. The do-nothing default:

```
DECLARE
  zone_id INTEGER := :zoneid;
BEGIN
  NULL;
END;
```

A.4.2.5 Master/Slave Stuff

oracle-unfresh-zones-query Return a list of zones that need to be checked and their master servers. Return multiple rows, identical except for the master address, for zones with more than one master. Default:

```
SELECT z.id, z.name, z.last_check, z.serial, zm.master
FROM Zones z JOIN Zonemasters zm ON z.id = zm.zone_id
WHERE z.type = 'SLAVE'
  AND (z.last_check IS NULL OR z.last_check + z.refresh < :ts)
ORDER BY z.id
```

oracle-zone-set-last-check-query Set the last check timestamp after a successful check. Default:

```
UPDATE Zones SET last_check = :lastcheck WHERE id = :zoneid
```

oracle-updated-masters-query Return a list of zones that need to have NOTIFY packets sent out. Default:

```
SELECT id, name, serial, notified_serial
FROM Zones
WHERE type = 'MASTER'
  AND (notified_serial IS NULL OR notified_serial < serial)
```

oracle-zone-set-notified-serial-query Set the last notified serial after packets have been sent. Default:

```
UPDATE Zones SET notified_serial = :serial WHERE id = :zoneid
```

oracle-also-notify-query Return a list of hosts that should be notified, in addition to any nameservers in the NS records, when sending NOTIFY packets for the named zone. Default:

```
SELECT an.hostaddr
FROM Zones z JOIN ZoneAlsoNotify an ON z.id = an.zone_id
WHERE z.name = lower(:name)
```

oracle-zone-masters-query Return a list of masters for the zone specified by id. Default:

```
SELECT master
FROM Zonemasters
WHERE zone_id = :zoneid
```

oracle-is-zone-master-query Return a row if the specified host is a registered master for the named zone. Default:

```
SELECT zm.master
FROM Zones z JOIN Zonemasters zm ON z.id = zm.zone_id
WHERE z.name = lower(:name) AND zm.master = :master
```

A.4.2.6 Superslave Stuff

oracle-accept-supernotification-query If a supernotification should be accepted from `:ip`, for the master nameserver `:ns`, return a label for this supermaster. Default:

```
SELECT name
FROM Supermasters
WHERE ip = :ip AND nameserver = lower(:ns)
```

oracle-insert-slave-query A supernotification has just been accepted, and we need to create an entry for the new zone. Default:

```
INSERT INTO Zones (id, name, type)
VALUES (zones_id_seq.NEXTVAL, lower(:zone), 'SLAVE')
RETURNING id INTO :zoneid
```

oracle-insert-master-query We need to register the first master server for the newly created zone. Default:

```
INSERT INTO Zonemasters (zone_id, master)
VALUES (:zoneid, :ip)
```

A.5 Generic SQLite backend (2 and 3)



Warning

When importing large amounts of data, be sure to run 'analyze;' afterwards as SQLite3 has a tendency to use sub-optimal indexes otherwise.

Native	Yes
Master	Yes
Slave	Yes
Superslave	Yes
DNSSEC	gsqli3 only (set gsqli3-dnssec)
Module name	gsqli3 and gsqli3
Launch name	gsqli3 and gsqli3

Table A.5: Generic SQLite backend capabilities

This backend retrieves all data from a SQLite database, which is an RDBMS that's embedded into the application itself, so you won't need to be running a separate server process. It also reduces overhead, and simplifies installation. At <http://www.sqlite.org> you can find more information about SQLite.

As this is a generic backend, built on top of the gSql framework, you can specify all queries as documented in [Generic MySQL and PostgreSQL backends](#).

SQLite exists in two incompatible versions, numbered 2 and 3, and from 2.9.21 onwards, PowerDNS supports both. It is recommended to go with version 3 as it is newer, has better performance and is actively maintained. To use version 3, choose 'launch=gsqli3'.

A.5.1 Compiling the SQLite backend

Before you can begin compiling PowerDNS with the SQLite backend you need to have the SQLite utility and library installed on your system. You can download these from <http://www.sqlite.org/download.html>, or you can use packages (if your distribution provides those).

When you've installed the library you can use: `./configure --with-modules="gsqli3"` or `./configure --with-modules="gsqli3"` to configure PowerDNS to use the SQLite backend. Compilation can then proceed as usual.

SQLite is included in most PowerDNS binary releases.

A.5.2 Setting up the database

Before you can use this backend you first have to set it up and fill it with data. The default setup conforms to the following schema:

```
create table domains (
  id          INTEGER PRIMARY KEY,
  name       VARCHAR(255) NOT NULL COLLATE NOCASE,
  master     VARCHAR(128) DEFAULT NULL,
  last_check INTEGER DEFAULT NULL,
  type       VARCHAR(6) NOT NULL,
  notified_serial INTEGER DEFAULT NULL,
  account    VARCHAR(40) DEFAULT NULL
);
```

```
CREATE UNIQUE INDEX name_index ON domains(name);
```

```
CREATE TABLE records (
  id          INTEGER PRIMARY KEY,
  domain_id   INTEGER DEFAULT NULL,
  name       VARCHAR(255) DEFAULT NULL,
  type       VARCHAR(10) DEFAULT NULL,
  content     VARCHAR(65535) DEFAULT NULL,
  ttl        INTEGER DEFAULT NULL,
  prio       INTEGER DEFAULT NULL,
  change_date INTEGER DEFAULT NULL
```

```
);

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 COLLATE NOCASE,
  account     VARCHAR(40) DEFAULT NULL
);
```

This schema contains all elements needed for master, slave and superslave operation.

To support DNSSEC, or to migrate to DNSSEC, the following statements must be issued:

```
alter table records add ordername      VARCHAR(255);
alter table records add auth bool;
create index orderindex on records(ordername);

create table domainmetadata (
  id      INTEGER PRIMARY KEY,
  domain_id  INT NOT NULL,
  kind     VARCHAR(16) COLLATE NOCASE,
  content  TEXT
);

create index domainmetaidindex on domainmetadata(domain_id);

create table cryptokeys (
  id      INTEGER PRIMARY KEY,
  domain_id  INT NOT NULL,
  flags    INT NOT NULL,
  active   BOOL,
  content  TEXT
);

create index domainidindex on cryptokeys(domain_id);

create table tsigkeys (
  id      INTEGER PRIMARY KEY,
  name    VARCHAR(255) COLLATE NOCASE,
  algorithm VARCHAR(50) COLLATE NOCASE,
  secret  VARCHAR(255)
);

create unique index namealgoindex on tsigkeys(name, algorithm);
```

For full migration notes, please see Section [12.3](#).

After you have created the database you probably want to fill it with data. If you have a BIND zone file it's as easy as: **zone2sql --zone=myzonefile --mysql | sqlite powerdns.sqlite**, but you can also use AXFR (or insert data manually).

To communicate with a SQLite database, use either the 'sqlite' or 'sqlite3' program, and feed it SQL.

A.5.3 Using the SQLite backend

The last thing you need to do is telling PowerDNS to use the SQLite backend.

```
# in pdns.conf
launch=sqlite # or sqlite3
sqlite-database=<path to your SQLite database> # or sqlite3-database
```

Then you can start PowerDNS and it should notify you that a connection to the database was made.

A.6 DB2 backend

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

Table A.6: DB2 backend capabilities

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.7 Bind zone file backend

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 'work-alike'.

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. This feature is no longer supported from 2.9.21 onwards.

bind-config= Location of the Bind configuration file to parse.

bind-check-interval= How often to check for zone changes. See 'Operation' section.

Native	Yes
Master	Yes
Slave	Yes
Superslave	Experimental
Autoserial	No
DNSSEC	Yes, but no key storage
Module name	none (built in)
Launch	bind

Table A.7: Bind zone file backend capabilities

A.7.1 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.7.2 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.7.3 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.7.4 Master/slave configuration

A.7.4.1 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.7.4.2 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.7.5 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.

reload All zones with a changed timestamp are reloaded at the next incoming query for them.

A.8 ODBC backend

Note

This backend was removed in version 3.1.

Native	Yes
Master	Yes (experimental)
Slave	Yes (experimental)
Superslave	No
Autoserial	Yes

Table A.8: ODBC backend capabilities

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(10) 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 data source.

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.9 XDB Backend

No longer part of PowerDNS.

A.10 LDAP backend



Warning

As of PowerDNS Authoritative Server 3.0, the LDAP backend is unmaintained. While care will be taken that this backend still compiles, this backend is known to have problems in version 3.0 and beyond! Please contact powerdns.support@netherlabs.nl or visit www.powerdns.com to rectify this situation.



Warning

This documentation has moved to [its own page](#). The information in this chapter may be outdated!

The main author for this module is Norbert Sendetzky.

He also maintains the [LDAP backends documentation](#) there. The information below may be outdated!



Warning

Host names and the MNAME of a SOA records are NEVER terminated with a '.' in PowerDNS storage! If a trailing '.' is present it will inevitably cause problems, problems that may be hard to debug.

Native	Yes
Master	No
Slave	No
Superslave	No
Autoserial	No
DNSSEC	No

Table A.9: LDAP backend capabilities

A.11 OpenDBX backend



Warning

The full OpenDBX documentation can be found on [its own page](#). The information in this chapter may be outdated!

The main author for this module is Norbert Sendetzky.

Native	Yes
Master	Yes
Slave	Yes
Superslave	Yes
Autoserial	Yes (since 2.9.22)
DNSSEC	No

Table A.10: OpenDBX backend capabilities

A.12 Geo backend



Warning

This section is a subset of the full documentation which can be found in `modules/geobackend/README` of the PowerDNS distribution.

The main author for this module is Mark Bergsma.

Native	Partial
Master	No
Slave	No
Superslave	No
Autoserial	No
DNSSEC	Yes (no key storage)

Table A.11: Geo backend capabilities

The Geo Backend can be used to distribute queries globally using an IP-address/country mapping table, several of which are freely available online or can be acquired for a small fee.

This allows visitors to be sent to a server close to them, with no appreciable delay, as would otherwise be incurred with a protocol level redirect. Additionally, the Geo Backend can be used to provide service over several clusters, any of which can be taken out of use easily, for example for maintenance purposes.

The Geo Backend is in wide use, for example by the Wikimedia foundation, which uses it to power the Wikipedia global load balancing.

More details can be found [here](#), or in `modules/geobackend/README`, part of the PowerDNS Authoritative Server distribution.

A.13 MongoDB Backend



Warning

This section is a subset of the full documentation which can be found in `modules/mongodbbbackend/README` of the PowerDNS distribution.

The main author for this module is Fredrik Danerklint.

Native	Yes
Master	Yes
Slave	No
Superslave	No
Autoserial	No
DNSSEC	Yes

Table A.12: MongoDB backend capabilities



Warning

The MongoDB Backend is available since PowerDNS Authoritative Server 3.0. In 3.0 and 3.1, this backend is marked as Experimental!

The MongoDB backend is a full service backend that stores data in a MongoDB instance.

More details can be found [here](#), or in `modules/mongodbbbackend/README`, part of the PowerDNS Authoritative Server distribution.

A.14 Lua Backend



Warning

This section is a subset of the full documentation which can be found in `modules/luabackend/README` of the PowerDNS distribution.

The main author for this module is Fredrik Danerklint.

Native	Yes
Master	Yes
Slave	No
Superslave	No
Autoserial	No
DNSSEC	Yes

Table A.13: Lua backend capabilities



Warning

The Lua Backend is available since PowerDNS Authoritative Server 3.0. In 3.0 and 3.1, this backend is marked as Experimental!

The Lua backend is a full service that can allow a Lua script to provide answers to DNS queries.

More details can be found [here](#), or in `modules/luabackend/README`, part of the PowerDNS Authoritative Server distribution.

A.15 TinyDNS Backend



Warning

The TinyDNS Backend is available since PowerDNS Authoritative Server 3.1. This backend is marked as experimental!

Native	Yes
Master	Yes
Slave	No
Superslave	No
Autoserial	No
DNSSEC	No
Multiple instances	Yes

Table A.14: TinyDNS backend capabilities

The TinyDNS backend allows you to use `djbdns's` `data.cdb` file format as the storage of your DNS records. The `data.cdb` file is created using `tinydns-data`. The backend is designed to be able to use the `data.cdb` files without any changes.

A.15.1 Configuration Parameters

These are the configuration file parameters that are available for the TinyDNS backend. It is recommended to set the `tinydns-dbfile`.

tinydns-dbfile Specifies the name of the data file to use. The default is `'data.cdb'`.

tinydns-tai-adjust This adjusts the **TAI** value if timestamps are used. These seconds will be added to the start point (1970) and will allow you to adjust for leap seconds. The current default is 10, **but as of June 30th 2012** should be 11.

tinydns-notify-on-startup Tell the TinyDNSBackend to notify all the slave nameservers on startup. This might cause broadcast storms. Default is no.

tinydns-locations Enable or Disable location support in the backend. Changing the value to `'no'` will make the backend ignore the locations. This then returns all records. When the setting is changed to `'no'` an AXFR will also return all the records. With the setting on `'yes'` an AXFR will only return records without a location.

A.15.2 Location and Timestamp support

Both timestamp and location are supported in the backend. Locations support can be changed using the **tinydns-locations** setting. Timestamp and location only work as expected when **cache-ttl** and **query-cache-ttl** are set to 0 (which disables these caches). Timestamp can operate with **cache-ttl** if cache is needed, but the TTL returned for the timestamped record will not be totally correct. The record will expire once the cache is expired and the backend is queried again. Please note that **cache-ttl** is a performance related setting. See Section 9.3. Location support only exists for IPv4!

A.15.3 Master mode

The TinyDNSBackend supports master mode. This allows it to notify slave nameservers of updates to a zone. You simply need to rewrite the data.cdb file with an updated/increased serial and PowerDNS will notify the slave nameservers of that domain. The **tinydns-notify-on-startup** configuration setting tells the backend if it should notify all the slave nameservers just after startup.

The CDB datafile does not allow PowerDNS to easily query for newly added domains or updated serial numbers. The CDB datafile requires us to do a full scan of all the records. When running with verbose logging, this could lead to a lot of output. The scanning of the CDB file may also take a while on systems with large files. The scan happens at an interval set by the **slave-cycle-interval**. It might be useful to raise this value to limit the amount of scans on the CDB file.

The TinyDNSBackend also keeps a list of all the zones. This is needed to detect an updated serial and to give every zone a unique id. The list is updated when a zone is added, but not when a zone is removed. This leads to some memory loss.

A.15.4 Useful implementation notes

This backend might solve some issues you have with the current tinydns noted on [Jonathan de Boyne Pollard's djbdns known problems page](#).

The data.cdb file format support all types of records. They are sometimes difficult to create because you need to specify the actual content of the rdata. [Tinydns.org](#) provides a number of links to tools/cgi-scripts that allow you to create records. [Anders Brownworth](#) also provides a number of useful record building scripts on his [djbdnsRecordBuilder](#).

Compiling the TinyDNS backend requires you to have [tinycdb](#) version 0.77.

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 18.3.

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.

`cycle` Restart a PowerDNS instance. Only available when running in guardian mode.

`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.

`ping` 'PING' the powerdns-guardian. Will return 'PONG' when it is available. (Only works when you are running in guardian mode)

`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.

`rping` 'PING' the powerdns-instance. Will return 'PONG' when it is available.

- 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 immediately.
- 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.
- show variable** Show a specific statistic. Use * for all. (You may need to quote as '*' or '*').
- version** Returns the version of a running `pdns` daemon.
- status** Retrieves the status of PowerDNS. Only available when running with guardian.

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 command line, 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 if 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.

PDNS breaks strict RFC compatibility 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. Before doing any PowerDNS development, please visit [the wiki](#).

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.

**Warning**

PowerDNS may instantiate multiple instances of your backend, or destroy existing copies and instantiate new ones. Backend code should therefore be thread-safe with respect to its static data. Additionally, it is wise if instantiation is a fast operation, with the possible exception of the first construction.

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 void lookup(const QType &qtype, const string &qdomain, DNSPacket *pkt_p=0, int ↵
    zoneId=-1)=0;
virtual bool list(const string &target, int domain_id)=0;
virtual bool get(DNSResourceRecord &r)=0;
virtual bool getSOA(const string &name, SOAData &soadata, DNSPacket *p=0);
};
```

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 **DNSResourceRecords** 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, there 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 19) 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(const string &target, 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 ↵
            random.powerdns.com A
            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 ←
        answer

        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:
    RandomLoader()
    {
        BackendMakers().report(new RandomFactory);

        L<<Logger::Info<<" [RandomBackend] This is the randombackend ("__DATE__", "__TIME__") ←
            reporting"<<endl;
    }
};

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!

C.1.2 Interface definition

Classes:

QType qtype	QType of this record
string qname	name of this record
string content	ASCII representation of right hand side
uint16_t priority	priority of an MX record.
uint32_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
bool auth	Used for DNSSEC operations. See Chapter 12 and more specifically Section 12.3. It is also useful to check out the <code>rectifyZone()</code> in <code>pdnssec.cc</code>

Table C.1: DNSResourceRecord class

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

Table C.2: SOAData struct

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 successfully, as indicated by returning **true**, answers should be made available over the `get()` method.

Should throw an `AhuException` if an error occurred accessing the database. Returning otherwise indicates that the query was started successfully. 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 occurred 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()` or `list()`. If this function 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 data for `'case.yourdomain.com'`, the `qname` field of `rr` should contain `'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 command line. 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 ¶m, 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. They 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 if 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 **SlaveDomains** 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 zone transfer 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);
    /* ... */
}
```

uint32_t id	ID of this zone within this backend
string master	IP address of the master of this domain, if any
uint32_t serial	Serial number of this zone
uint32_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

Table C.3: DomainInfo struct

The mentioned DomainInfo struct looks like this:

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, ←
        const vector<DNSResourceRecord>&nssset, string *account, DNSBackend **db)
};
```

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 [18.2.1](#).

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 setNotified(uint32_t id, uint32_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(uint32_t domain_id, uint32_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](#). `./configure ; make ; make install` will do The Right Thing for most people.

PowerDNS can be compiled 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.

Starting with version 2.9.18, PowerDNS requires 'Boost' to compile, it is available for most operating systems. Otherwise, see [the Boost website](#).

If your operating system does not have a Boost package, you don't need to compile all of boost just for PowerDNS. PowerDNS only uses Boost include files, so there is no need to install all of boost. Just untar the Boost distribution file and point instruct `./configure` to find it, perhaps like this:

```
$ CXXFLAGS=-I/home/bert/download/boost_1_33_0 ./configure ...
```

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.

The FreeBSD Boost include files are installed in `/usr/local/include`, so prefix `CXXFLAGS=-I/usr/local/include` to your `./configure` invocation.

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 very well. 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.

When compiling, make sure that you have `/usr/ccs/bin` in your path. Furthermore, with some versions of MySQL, you may have to add "LDFLAGS=-lz" before `./configure`.

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 were 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. Throughout 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 Prerequisites

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.2.2.1.2 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](#).

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\Platform 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`).
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](#) 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 wonderful 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

Microsoft, Visual C++, Windows, Windows NT, Windows 2000, Windows XP and Win32 are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

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

Appendix F

Further copyright statements

F.1 AES implementation by Brian Gladman

Since version 3.1.5, PowerDNS contains AES code by Brian Gladman, to which the following applies:

Copyright © 1998-2007, Brian Gladman, Worcester, UK. All rights reserved.

LICENSE TERMS

The free distribution and use of this software is allowed (with or without changes) provided that:

1. source code distributions include the above copyright notice, this list of conditions and the following disclaimer;
2. binary distributions include the above copyright notice, this list of conditions and the following disclaimer in their documentation;
3. the name of the copyright holder is not used to endorse products built using this software without specific written permission.

DISCLAIMER

This software is provided 'as is' with no explicit or implied warranties in respect of its properties, including, but not limited to, correctness and/or fitness for purpose.

Appendix G

Cryptographic software and export control

In certain legal climates, PowerDNS might potentially require an export control status, particularly since PowerDNS software contains cryptographic primitives.

PowerDNS does not itself implement any cryptographic algorithms but relies on third party implementations of AES, RSA, ECDSA, GOST, MD5 and various SHA-based hashing algorithms.

For AES, we rely on Brian Gladman's code, as outlined in Appendix F. Furthermore, RSA, MD5 and the SHA-based algorithms are supplied as a copy of [PolarSSL](#).

Optionally, PowerDNS can link in a copy of the open source [Botan cryptographic](#) library.

Optionally, PowerDNS can link in a copy of the open source [Crypto++](#) library.

G.1 Specific United States Export Control Notes

PowerDNS is not "US Origin" software. For re-export, like most open source, publicly available "mass market" projects, PowerDNS is considered to be governed by section 740.13(e) of the US EAR, "Unrestricted encryption source code", under which PowerDNS source code would be considered re-exportable from the US without an export license under License Exception TSU (Technology and Software - Unrestricted).

Like most open source projects containing some encryption, the ECCN that best fits PowerDNS software is 5D002.

The official link to the publicly available source code is <http://downloads.powerdns.com/releases>.

If absolute certainty is required, we recommend consulting an expert in US Export Control, or asking the BIS for confirmation.