



PowerDNS Authoritative Server Documentation

PowerDNS.COM BV

Mar 02, 2018

CONTENTS

1	PowerDNS Authoritative Nameserver	1
1.1	Getting Started	1
1.2	Getting Support	1
1.2.1	My information is confidential, must I send it to the mailing list or discuss on IRC? . . .	1
1.2.2	I have a question!	2
1.2.3	What details should I supply?	2
1.2.4	I found a bug!	2
1.2.5	I found a security issue!	2
1.2.6	I have a good idea for a feature!	2
2	Installing PowerDNS	3
2.1	Binary Packages	3
2.1.1	Debian-based Systems	3
2.1.2	Redhat-based Systems	3
2.1.3	FreeBSD	3
2.1.4	Mac OS X	4
2.2	After installation	4
3	Upgrade Notes	5
3.1	4.1.0 to 4.1.1	5
3.2	4.0.X to 4.1.0	5
3.2.1	Changed options	5
3.2.2	Other changes	6
3.3	4.0.X to 4.0.2	6
3.3.1	Changed options	6
3.4	3.4.X to 4.0.0	6
3.4.1	Database changes	6
3.4.2	Changed options	6
3.4.3	API	7
3.4.4	Resource Record Changes	7
4	DNS Modes of Operation	9
4.1	Native replication	9
4.2	Master operation	9
4.3	Slave operation	10
4.4	Master/Slave Setup Requirements	11
4.5	IXFR: incremental zone transfers	11
4.6	Supermaster: automatic provisioning of slaves	11
4.7	Modifying a slave zone using a script	12
5	Migrating to PowerDNS	15
5.1	Using AXFR to a Slave-Capable Backend	15
5.1.1	To A Generic SQL Backend	15
5.1.2	To the BIND backend	15
5.2	From zonefiles to PowerDNS	16

5.2.1	Using the BIND backend	16
5.2.2	To a Generic SQL backend	16
5.3	Migrating Data from one Backend to Another Backend	17
5.3.1	Prerequisites	17
5.3.2	Moving from source to target	17
6	Running and Operating	19
6.1	Guardian	19
6.2	Logging to syslog on systemd-based operating systems	19
6.3	Logging to syslog	19
6.4	Controlling A Running PowerDNS Server	20
6.4.1	Control Socket	20
6.4.2	pdns_control	20
6.5	The SysV init script	20
6.6	Running in the foreground	21
7	Security of PowerDNS	23
7.1	PowerDNS Security Policy	23
7.1.1	HackerOne	23
7.1.2	Disclosure Policy	23
7.2	Securing the Process	24
7.2.1	Running as a less privileged identity	24
7.2.2	Jailing the process in a chroot	24
7.3	Security Considerations	24
7.4	Security Polling	24
7.4.1	Details	25
8	Performance and Tuning	27
8.1	Performance related settings	27
8.2	Packet Cache	28
8.3	Query Cache	28
8.4	Performance Monitoring	28
8.4.1	Counters	28
8.4.2	Ring buffers	33
8.4.3	Sending metrics to Graphite/Metronome over Carbon	34
9	DNSSEC	35
9.1	A brief introduction to DNSSEC	35
9.2	DNSSEC Profile and Support	36
9.2.1	Supported Algorithms	36
9.3	DNSSEC Modes of Operation	37
9.3.1	Online Signing	37
9.3.2	Pre-signed records	38
9.3.3	Front-signing	39
9.3.4	Signed AXFR	39
9.3.5	BIND-mode operation	39
9.3.6	Hybrid BIND-mode operation	39
9.4	pdnsutil and DNSSEC	40
9.4.1	DNSSEC Defaults	40
9.5	Migrating (Signed) Zones to PowerDNS	40
9.5.1	From an existing PowerDNS installation	40
9.5.2	From existing non-DNSSEC, non-PowerDNS setups	41
9.5.3	From existing DNSSEC non-PowerDNS setups, pre-signed	41
9.5.4	From existing DNSSEC non-PowerDNS setups, live signing	41
9.5.5	Secure transfers	41
9.6	Operational instructions	41
9.6.1	Publishing a DS	42
9.6.2	Going insecure	42
9.6.3	Setting the NSEC modes and parameters	42

9.6.4	SOA-EDIT: ensure signature freshness on slaves	42
9.6.5	Security	44
9.6.6	Performance	44
9.7	DNSSEC advice & precautions	44
9.7.1	Packet sizes, fragments, TCP/IP service	45
9.8	PKCS#11 support	45
9.8.1	Using with SoftHSM	45
9.8.2	Using CryptAS	46
9.9	Thanks to, acknowledgements	47
10	Per zone settings: Domain Metadata	49
10.1	ALLOW-AXFR-FROM	49
10.2	API-RECTIFY	49
10.3	AXFR-SOURCE	50
10.4	ALLOW-DNSUPDATE-FROM, TSIG-ALLOW-DNSUPDATE, FORWARD-DNSUPDATE, SOA-EDIT-DNSUPDATE, NOTIFY-DNSUPDATE	50
10.5	ALSO-NOTIFY	50
10.6	AXFR-MASTER-TSIG	50
10.7	GSS-ALLOW-AXFR-PRINCIPAL	50
10.8	GSS-ACCEPTOR-PRINCIPAL	50
10.9	IXFR	50
10.10	LUA-AXFR-SCRIPT	51
10.11	NSEC3NARROW	51
10.12	NSEC3PARAM	51
10.13	PRESIGNED	51
10.14	PUBLISH-CDNSKEY, PUBLISH-CDS	51
10.15	SOA-EDIT	51
10.16	SOA-EDIT-API	51
10.17	TSIG-ALLOW-AXFR	52
10.18	TSIG-ALLOW-DNSUPDATE	52
10.19	Extra metadata	52
11	Dynamic DNS Update (RFC2136)	53
11.1	Configuration options	53
11.1.1	dnsupdate	53
11.1.2	allow-dnsupdate-from	53
11.1.3	forward-dnsupdate	53
11.1.4	lua-dnsupdate-policy-script	54
11.2	Per zone settings	54
11.2.1	ALLOW-DNSUPDATE-FROM	54
11.2.2	TSIG-ALLOW-DNSUPDATE	54
11.2.3	FORWARD-DNSUPDATE	55
11.2.4	NOTIFY-DNSUPDATE	55
11.2.5	SOA-EDIT-DNSUPDATE	55
11.3	SOA Serial Updates	55
11.3.1	SOA-EDIT-DNSUPDATE settings	56
11.4	DNS update How-to: Setup dyndns/rfc2136 with dhcpd	56
11.4.1	Setting up dhcpd	56
11.4.2	Setting up PowerDNS	57
11.5	How it works	58
11.6	Update policy	59
12	TSIG	61
12.1	Provisioning outbound AXFR access	61
12.2	Provisioning signed notification and AXFR requests	62
12.3	GSS-TSIG support	62
12.3.1	Prerequisites	63
12.3.2	Setting up	63

13	Guides and How Tos	65
13.1	Basic setup: configuring database connectivity	65
13.1.1	Example: configuring MySQL	65
13.1.2	Common problems	69
13.1.3	Typical Errors after Installing	69
13.2	Migrating from using recursion on the Authoritative Server to using a Recursor	70
13.2.1	Scenario 1: Authoritative Server as Recursor with private zones	70
13.2.2	Scenario 2: Authoritative Server as Recursor for clients and serving public domains	71
13.3	Running Virtual Instances	73
13.3.1	Starting virtual instances with Sysv init-scripts	73
13.3.2	Starting virtual instances with systemd	74
13.4	Using ALIAS records	74
13.4.1	ALIAS and DNSSEC	74
13.5	KSK Rollover	75
13.6	KSK Rollover using CDS & CDNSKEY Key Rollover	75
13.7	ZSK Rollover	76
13.8	Adding new DNS record types	76
14	Backends	79
14.1	Bind zone file backend	79
14.1.1	Configuration Parameters	80
14.1.2	Operation	80
14.1.3	pdns_control commands	81
14.1.4	Performance	81
14.1.5	Master/slave/native configuration	81
14.2	Generic SQL Backends	82
14.2.1	Basic functionality	82
14.2.2	Handling DNSSEC signed zones	84
14.2.3	Queries	84
14.3	Generic MySQL backend	87
14.3.1	Using MySQL replication	88
14.3.2	Settings	88
14.3.3	Default Schema	89
14.4	Generic ODBC Backend	91
14.4.1	Enabling the backend	91
14.4.2	Configuration Parameters	91
14.4.3	Connecting to Microsoft SQL Server	92
14.5	Generic Oracle backend	95
14.5.1	Settings	97
14.5.2	Caveats	97
14.6	Generic PostgreSQL backend	97
14.6.1	Settings	97
14.6.2	Default schema	98
14.7	Generic SQLite 3 backend	100
14.7.1	Setting up the database	100
14.7.2	Configuration Parameters	102
14.7.3	Using the SQLite backend	103
14.7.4	Compiling the SQLite backend	103
14.8	GeoIP backend	103
14.8.1	Prerequisites	103
14.8.2	Configuration Parameters	104
14.8.3	Zonefile format	104
14.9	LDAP backend	107
14.9.1	Introduction	107
14.9.2	Installation	112
14.9.3	Configuration options	112
14.9.4	Master Mode	114
14.9.5	Example	115

14.9.6	Migration	117
14.9.7	Optimization	118
14.9.8	Troubleshooting	119
14.9.9	Future	119
14.10	Lua Backend	120
14.10.1	New functions	120
14.10.2	What has been tested	121
14.10.3	What you will find under the test directory	121
14.10.4	Parameters	121
14.10.5	DNSSEC	122
14.10.6	Information for logging	122
14.11	MyDNS Backend	122
14.11.1	Configuration Parameters	123
14.11.2	Migrating from MyDNS to another SQL backend	124
14.12	OpenDBX Backend	124
14.12.1	Settings	124
14.12.2	Queries	125
14.12.3	Database schemas and information	126
14.13	Oracle backend	137
14.13.1	Configuration Parameters	137
14.13.2	The Database Schema	138
14.14	Pipe Backend	147
14.14.1	Configuration Parameters	148
14.14.2	PipeBackend protocol	148
14.14.3	Sample backends	151
14.15	Random Backend	155
14.15.1	Configuration Parameters	156
14.16	Remote Backend	156
14.16.1	Important notices	156
14.16.2	Compiling	156
14.16.3	Usage	157
14.16.4	API	157
14.16.5	Examples	178
14.17	TinyDNS Backend	179
14.17.1	Configuration Parameters	180
14.17.2	Location and Timestamp support	180
14.17.3	Master mode	181
14.17.4	Useful implementation Notes	181
15	Built-in Webserver and HTTP API	183
15.1	Webserver	183
15.2	Enabling the API	183
15.3	Working with the API	184
15.3.1	Authentication	184
15.3.2	Errors	184
15.3.3	Data format	184
15.4	Endpoints and Objects in the API	184
15.4.1	Servers	184
15.4.2	Zones	185
15.4.3	Cryptokeys	189
15.4.4	Metadata	191
15.4.5	Searching	192
15.4.6	Statistics	193
16	Manual Pages	195
16.1	calidns	195
16.1.1	Synopsis	195
16.1.2	Description	195

16.1.3	QUERY_FILE format	195
16.1.4	Options	195
16.2	dnsbulktest	196
16.2.1	Synopsis	196
16.2.2	Description	196
16.2.3	Options	196
16.3	dnsgram	196
16.3.1	Synopsis	196
16.3.2	Description	196
16.3.3	Options	196
16.3.4	See also	196
16.4	dnspcap2protobuf	196
16.4.1	Synopsis	196
16.4.2	Description	197
16.4.3	Options	197
16.5	dnsreplay	197
16.5.1	Synopsis	197
16.5.2	Description	197
16.5.3	Options	197
16.5.4	Bugs	197
16.5.5	See also	198
16.6	dnsscan	198
16.6.1	Synopsis	198
16.6.2	Description	198
16.6.3	Options	198
16.6.4	See also	198
16.7	dnsscope	198
16.7.1	Synopsis	198
16.7.2	Description	198
16.7.3	Options	198
16.7.4	See also	199
16.8	dnstcpbench	199
16.8.1	Synopsis	199
16.8.2	Description	199
16.8.3	Options	199
16.8.4	Bugs	199
16.9	dnswasher	199
16.9.1	Synopsis	199
16.9.2	Description	200
16.9.3	Options	200
16.9.4	See also	200
16.10	dumresp	200
16.10.1	Synopsis	200
16.10.2	Description	200
16.10.3	Options	200
16.10.4	See also	200
16.11	ixfrdist	200
16.11.1	Synopsis	200
16.11.2	Description	201
16.11.3	Options	201
16.11.4	See also	201
16.12	ixplore	201
16.12.1	Synopsis	201
16.12.2	Description	201
16.12.3	Options	202
16.12.4	diff-mode	202
16.12.5	track-mode	202
16.12.6	See also	202

16.13	nproxy	202
16.13.1	Synopsis	202
16.13.2	Description	202
16.13.3	Options	203
16.14	nsec3dig	203
16.14.1	Synopsis	203
16.14.2	Description	203
16.14.3	Example	203
16.15	pdns_control	203
16.15.1	Synopsis	203
16.15.2	Description	203
16.15.3	Options	203
16.15.4	Commands	204
16.15.5	See also	206
16.16	pdns_notify	206
16.16.1	Synopsis	206
16.16.2	Description	206
16.16.3	Options	206
16.17	pdns_server	206
16.17.1	Synopsis	206
16.17.2	Description	207
16.17.3	Options	207
16.17.4	See also	207
16.18	pdnsutil	207
16.18.1	Synopsis	207
16.18.2	Description	207
16.18.3	Options	207
16.18.4	COMMANDS	207
16.18.5	DNSSEC RELATED COMMANDS	208
16.18.6	TSIG RELATED COMMANDS	209
16.18.7	ZONE MANIPULATION COMMANDS	209
16.18.8	DEBUGGING TOOLS	210
16.18.9	See also	210
16.19	saxfr	210
16.19.1	Synopsis	210
16.19.2	Description	210
16.19.3	Options	211
16.20	sdig	211
16.20.1	Synopsis	211
16.20.2	Description	211
16.20.3	Options	211
16.21	zone2json	211
16.21.1	Synopsis	211
16.21.2	Description	211
16.21.3	Options	211
16.21.4	INPUT Options	211
16.21.5	OTHER Options	212
16.21.6	See also	212
16.22	zone2ldap	212
16.22.1	Synopsis	212
16.22.2	Description	212
16.22.3	Options	212
16.22.4	See also	212
16.23	zone2sql	212
16.23.1	Synopsis	212
16.23.2	Description	213
16.23.3	Options	213
16.23.4	INPUT Options	213

16.23.5 BACKENDS	213
16.23.6 OUTPUT Options	213
16.23.7 OTHER Options	213
16.23.8 JSON COMMENTS	214
16.23.9 See also	214
17 Authoritative Server Settings	215
17.1 8bit-dns	215
17.2 allow-axfr-ips	215
17.3 allow-dnsupdate-from	215
17.4 allow-notify-from	215
17.5 allow-unsigned-notify	216
17.6 allow-unsigned-supermaster	216
17.7 allow-recursion	216
17.8 also-notify	216
17.9 any-to-tcp	216
17.10 api	217
17.11 api-key	217
17.12 api-readonly	217
17.13 axfr-lower-serial	217
17.14 cache-ttl	217
17.15 carbon-ourname	217
17.16 carbon-server	218
17.17 carbon-interval	218
17.18 chroot	218
17.19 config-dir	218
17.20 config-name	218
17.21 control-console	218
17.22 daemon	219
17.23 default-ksk-algorithm	219
17.24 default-ksk-size	219
17.25 default-soa-name	219
17.26 default-soa-edit	220
17.27 default-soa-edit-signed	220
17.28 default-soa-mail	220
17.29 default-ttl	220
17.30 default-zsk-algorithm	220
17.31 default-zsk-size	221
17.32 direct-dnskey	221
17.33 disable-axfr	221
17.34 disable-axfr-rectify	221
17.35 disable-syslog	221
17.36 disable-tcp	222
17.37 distributor-threads	222
17.38 dname-processing	222
17.39 dnssec-key-cache-ttl	222
17.40 dnsupdate	222
17.41 do-ipv6-additional-processing	222
17.42 domain-metadata-cache-ttl	223
17.43 edns-subnet-processing	223
17.44 entropy-source	223
17.45 expand-alias	223
17.46 forward-dnsupdate	223
17.47 forward-notify	223
17.48 guardian	224
17.49 include-dir	224
17.50 launch	224
17.51 load-modules	224

17.52	local-address	224
17.53	log-timestamp	225
17.54	non-local-bind	225
17.55	lua-axfr-script	225
17.56	local-address-nonexist-fail	225
17.57	local-ipv6	225
17.58	local-ipv6-nonexist-fail	226
17.59	local-port	226
17.60	log-dns-details	226
17.61	logging-facility	226
17.62	loglevel	226
17.63	log-dns-queries	226
17.64	lua-prequery-script	226
17.65	master	227
17.66	max-cache-entries	227
17.67	max-ent-entries	227
17.68	max-nsec3-iterations	227
17.69	max-packet-cache-entries	227
17.70	max-queue-length	227
17.71	max-signature-cache-entries	228
17.72	max-tcp-connection-duration	228
17.73	max-tcp-connections	228
17.74	max-tcp-connections-per-client	228
17.75	max-tcp-transactions-per-conn	228
17.76	module-dir	228
17.77	negquery-cache-ttl	228
17.78	no-config	229
17.79	no-shuffle	229
17.80	overload-queue-length	229
17.81	reuseport	229
17.82	security-poll-suffix	229
17.83	server-id	229
17.84	only-notify	230
17.85	out-of-zone-additional-processing	230
17.86	outgoing-axfr-expand-alias	230
17.87	prevent-self-notification	230
17.88	query-cache-ttl	231
17.89	query-local-address	231
17.90	query-local-address6	231
17.91	query-logging	231
17.92	queue-limit	231
17.93	receiver-threads	231
17.94	recursive-cache-ttl	232
17.95	recursor	232
17.96	resolver	232
17.97	retrieval-threads	232
17.98	setgid	232
17.99	setuid	232
17.100	slave	232
17.101	slave-cycle-interval	233
17.102	slave-renotify	233
17.103	signing-threads	233
17.104	soa-expire-default	233
17.105	soa-minimum-ttl	233
17.106	soa-refresh-default	233
17.107	soa-retry-default	234
17.108	socket-dir	234
17.109	tcp-control-address	234

17.110	tcp-control-port	234
17.111	tcp-control-range	234
17.112	tcp-control-secret	234
17.113	tcp-fast-open	234
17.114	tcp-idle-timeout	235
17.115	traceback-handler	235
17.116	trusted-notification-proxy	235
17.117	udp-truncation-threshold	235
17.118	version-string	235
17.119	webserver	235
17.120	webserver-address	236
17.121	webserver-allow-from	236
17.122	webserver-password	236
17.123	webserver-port	236
17.124	webserver-print-arguments	236
17.125	write-pid	236
17.126	xfr-max-received-mbytes	237
18	Security Advisories	239
18.1	PowerDNS Security Advisory 2008-02: By not responding to certain queries, domains become easier to spoof	239
18.2	PowerDNS Security Advisory 2008-03: Some PowerDNS Configurations can be forced to restart remotely	239
18.3	PowerDNS Security Advisory 2012-01: PowerDNS Authoritative Server can be caused to generate a traffic loop	240
18.4	PowerDNS Security Advisory 2015-01: Label decompression bug can cause crashes or CPU spikes	241
18.5	PowerDNS Security Advisory 2015-02: Packet parsing bug can cause thread or process abortion	242
18.6	PowerDNS Security Advisory 2015-03: Packet parsing bug can lead to crashes	242
18.7	PowerDNS Security Advisory 2016-01: Crafted queries can cause unexpected backend load	243
18.8	PowerDNS Security Advisory 2016-02: Crafted queries can cause abnormal CPU usage	244
18.9	PowerDNS Security Advisory 2016-03: Denial of service via the web server	244
18.10	PowerDNS Security Advisory 2016-04: Insufficient validation of TSIG signatures	245
18.11	PowerDNS Security Advisory 2016-05: Crafted zone record can cause a denial of service	245
18.12	PowerDNS Security Advisory 2017-04: Missing check on API operations	246
18.13	Older security advisories	246
19	Changelogs	249
19.1	Changelogs for 4.1.x	249
19.1.1	4.1.1	249
19.1.2	4.1.0	249
19.1.3	4.1.0-rc3	250
19.1.4	4.1.0-rc2	251
19.1.5	4.1.0-rc1	252
19.2	Changelogs for 4.0.x	256
19.2.1	PowerDNS Authoritative Server 4.0.5	256
19.2.2	PowerDNS Authoritative Server 4.0.4	257
19.2.3	PowerDNS Authoritative Server 4.0.3	259
19.2.4	PowerDNS Authoritative Server 4.0.2	259
19.2.5	PowerDNS Authoritative Server 4.0.1	260
19.2.6	PowerDNS Authoritative Server 4.0.0	261
19.2.7	PowerDNS Authoritative Server 4.0.0-rc2	262
19.2.8	PowerDNS Authoritative Server 4.0.0-beta1	263
19.2.9	PowerDNS Authoritative Server 4.0.0-alpha3	263
19.2.10	PowerDNS Authoritative Server 4.0.0-alpha2	264
19.2.11	PowerDNS Authoritative Server 4.0.0-alpha1	265
19.3	Changelogs for 3.x and older	265
19.3.1	PowerDNS Authoritative Server 3.4.9	265
19.3.2	PowerDNS Authoritative Server 3.4.8	265

19.3.3	PowerDNS Authoritative Server 3.4.7	266
19.3.4	PowerDNS Authoritative Server 3.4.6	266
19.3.5	PowerDNS Authoritative Server 3.3.3	267
19.3.6	PowerDNS Authoritative Server 3.4.5	267
19.3.7	PowerDNS Authoritative Server 3.3.2	267
19.3.8	PowerDNS Authoritative Server 3.4.4	268
19.3.9	PowerDNS Authoritative Server 3.4.3	269
19.3.10	PowerDNS Authoritative Server 3.4.2	270
19.3.11	PowerDNS Authoritative Server 3.4.1	271
19.3.12	PowerDNS Authoritative Server 3.4.0	272
19.3.13	PowerDNS Authoritative Server version 3.3.1	276
19.3.14	PowerDNS Authoritative Server version 3.3	277
19.3.15	PowerDNS Authoritative Server 3.2	281
19.3.16	PowerDNS Authoritative Server 3.1	286
19.3.17	Changes between RC1 and RC2	288
19.3.18	Authoritative Server version 2.9.22.6	292
19.3.19	Authoritative Server version 2.9.22.5	292
19.3.20	PowerDNS Authoritative Server 3.0.1	292
19.3.21	PowerDNS Authoritative Server 3.0	292
19.3.22	Authoritative Server version 2.9.22	297
19.3.23	Authoritative Server version 2.9.21.2	301
19.3.24	Authoritative Server version 2.9.21.1	301
19.3.25	PowerDNS Authoritative Server version 2.9.21	301
19.3.26	Version 2.9.20	304
19.3.27	Version 2.9.19	305
19.3.28	Version 2.9.18	307
19.3.29	Version 2.9.17	309
19.3.30	Version 2.9.16	310
19.3.31	Version 2.9.15	311
19.3.32	Version 2.9.14	312
19.3.33	Version 2.9.13	312
19.3.34	Version 2.9.12	313
19.3.35	Version 2.9.11	314
19.3.36	Version 2.9.10	314
19.3.37	Version 2.9.8	315
19.3.38	Version 2.9.7	315
19.3.39	Version 2.9.6	316
19.3.40	Version 2.9.5	317
19.3.41	Version 2.9.4	318
19.3.42	Version 2.9.3a	319
19.3.43	Version 2.9.2	321
19.3.44	Version 2.9.1	322
19.3.45	Version 2.9	323
19.3.46	Version 2.8	323
19.3.47	Version 2.7 and 2.7.1	323
19.3.48	Version 2.6.1	324
19.3.49	Version 2.6	324
19.3.50	Version 2.5.1	325
19.3.51	Version 2.5	325
19.3.52	Version 2.4	327
19.3.53	Version 2.3	327
19.3.54	Version 2.2	328
19.3.55	Version 2.1	329
19.3.56	Version 2.0.1	329
19.3.57	Version 2.0	330
19.3.58	Version 2.0 Release Candidate 2	330
19.3.59	Version 2.0 Release Candidate 1	331
19.3.60	Version 1.99.12 Prerelease	332

19.3.61	Version 1.99.11 Prerelease	332
19.3.62	Version 1.99.10 Prerelease	333
19.3.63	Version 1.99.9 Early Access Prerelease	333
19.3.64	Version 1.99.8 Early Access Prerelease	334
19.3.65	Version 1.99.7 Early Access Prerelease	335
19.3.66	Version 1.99.6 Early Access Prerelease	336
19.3.67	Version 1.99.5 Early Access Prerelease	336
19.3.68	Version 1.99.4 Early Access Prerelease	337
19.3.69	Version 1.99.3 Early Access Prerelease	338
19.3.70	Version 1.99.2 Early Access Prerelease	339
19.3.71	Version 1.99.1 Early Access Prerelease	340
20	End of life statements	341
20.1	PowerDNS Authoritative Server 3.x	341
20.2	PowerDNS Authoritative Server 2.x	341
21	Frequently Asked Questions	343
21.1	Replication	343
21.1.1	My PowerDNS Authoritative Server does not send NOTIFY messages	343
21.1.2	My PowerDNS Authoritative Server does not start AXFRs	343
21.1.3	Can PowerDNS Server act as Slave and Master at the same time?	343
21.1.4	How can I limit Zone Transfers (AXFR) per Domain?	343
21.1.5	I have a working Supermaster/Superslave setup but when I remove Domains from the Master they still remain on the Slave. Am I doing something wrong?	343
21.2	Operational	344
21.2.1	The ADDITIONAL is section different than BIND's answer, why?	344
21.2.2	My server is not answering with a verbose "ADDITIONAL SECTION" that includes A records for the nameservers of the domain queried	344
21.2.3	PowerDNS does not give authoritative answers, how come?	344
21.2.4	Master or Slave support is not working, PowerDNS is not picking up changes	344
21.2.5	My masters won't allow PowerDNS to access zones as it is using the wrong local IP address	344
21.2.6	PowerDNS does not answer queries on all my IP addresses (and I've ignored the warning I got about that at startup)	344
21.2.7	Linux Netfilter says your conntrack table is full?	344
21.3	Backends	345
21.3.1	Does PowerDNS support splitting of TXT records (multipart or multiline) with the MySQL backend?	345
21.3.2	I see this a lot of "Failed to execute mysql_query" or similar log-entries	345
21.3.3	Which backend should I use? There are so many!	345
21.3.4	Can I launch multiple backends simultaneously?	345
21.3.5	I've added extra fields to the domains and/or records table. Will this eventually affect the resolution process in any way?	345
21.3.6	Can I specify custom sql queries for the gmysql / gpgsql backend or are those hardcoded?	345
22	Backend writers' guide	347
22.1	Notes	347
22.2	Simple read-only native backends	347
22.3	A sample minimal backend	349
22.4	Interface definition	350
22.4.1	Classes	350
22.4.2	Methods	351
22.5	Reporting errors	352
22.6	Declaring and reading configuration details	352
22.7	Read/write slave-capable backends	353
22.8	Supermaster/Superslave capability	355
22.9	Read/write master-capable backends	355
22.10	DNS update support	356
22.11	DNS update support	357

22.12	Miscellaneous	358
22.12.1	ENT (Empty Non-Terminal)	358
23	Compiling PowerDNS	359
23.1	Getting the sources	359
23.2	Dependencies	359
23.3	Optional dependencies	360
23.3.1	ed25519 support with libsodium	360
23.3.2	ed25519 and ed448 support with libdecaf	360
23.3.3	systemd notify support	360
24	Cryptographic software and export control	361
24.1	Specific United States Export Control Notes	361
25	Internals	363
25.1	How PowerDNS translates DNS queries into backend queries	363
26	Supported Record Types	365
26.1	A	365
26.2	AAAA	365
26.3	AFSDB	365
26.4	ALIAS	365
26.5	CAA	366
26.6	CERT	366
26.7	CDNSKEY	366
26.8	CDS	366
26.9	CNAME	366
26.10	DNSKEY	366
26.11	DNAME	366
26.12	DS	366
26.13	HINFO	367
26.14	KEY	367
26.15	LOC	367
26.16	MX	367
26.17	NAPTR	367
26.18	NS	367
26.19	NSEC, NSEC3, NSEC3PARAM	367
26.20	OPENPGPKEY	367
26.21	PTR	368
26.22	RP	368
26.23	RRSIG	368
26.24	SOA	368
26.25	SPF	368
26.26	SSHFP	369
26.27	SRV	369
26.28	TKEY, TSIG	369
26.29	TLSA	369
26.30	SMIMEA	369
26.31	TXT	369
26.32	URI	369
26.33	Other types	370
	HTTP Routing Table	371
	Index	373

POWERDNS AUTHORITATIVE NAMESERVER

The PowerDNS Authoritative Server 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.

PowerDNS has the concepts of ‘backends’. A backend is a datastore that the server will consult that contains DNS records (and some meta-data). The backends range from database backends (*MySQL*, *PostgreSQL*, *Oracle*) and *Bind-zonefiles* to *co-processes* and *JSON API’s*.

Multiple backends can be enabled in the configuration by using the *launch* option. Each backend can be configured separately.

See the *backend* documentation for more information.

This documentation is also available as a [PDF document](#).

1.1 Getting Started

- *Install the Authoritative Server*
- *Configure the Server*
- *Configure the backend(s)*

1.2 Getting Support

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).

Public support is available via several different channels:

- This documentation
- [The mailing list](#)
- [#powerdns](#) on [irc.oftc.net](#)

The PowerDNS company can provide help or support you in private as well. For first class and rapid support, please contact powerdns.support@powerdns.com, or see the [.com website](#).

1.2.1 My information is confidential, must I send it to the mailing list or discuss on IRC?

Yes, we have a support policy called “Open Source Support: out in the open”.

If you desire privacy, please consider entering a support relationship with us, in which case we invite you to contact powerdns.support.sales@powerdns.com.

1.2.2 I have a question!

This happens, we're here to help! Read below on how you can get help

1.2.3 What details should I supply?

Start out with stating what you think should be happening. Quite often, wrong expectations are the actual problem. Furthermore, your operating system, which version of PowerDNS you use and where you got it from (RPM, .DEB, tar.bz2). If you compiled it yourself, what were the `./configure` parameters.

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

1.2.4 I found a bug!

As much as we'd like to think we are perfect, bugs happen. If you have found a bug, please file a bug report on [GitHub](#). Please fill in the template and we'll try our best to help you.

1.2.5 I found a security issue!

Please report this in private, see the *PowerDNS Security Policy*.

1.2.6 I have a good idea for a feature!

We like to work on new things! You can file a feature request on [GitHub](#).

INSTALLING POWERDNS

Installation of the PowerDNS Authoritative server on UNIX systems can be done in several ways:

- Binary packages provided by your distribution
- Binary packages provided by PowerDNS on repo.powerdns.com

2.1 Binary Packages

2.1.1 Debian-based Systems

PowerDNS Authoritative Server is available through the `apt` system.

```
# apt-get install pdns-server
```

Debian splits the backends into several different packages, install the required backend as follows:

```
# apt-get install pdns-backend-$backend
```

2.1.2 Redhat-based Systems

On RedHat based system there are 2 options to install PowerDNS, from EPEL, the repository from Kees Monshouwer or from the PowerDNS repositories:

Add either to your list of repositories and install PowerDNS by issuing:

```
# yum install pdns
```

The different backends can be installed using

```
# yum install pdns-backend-$backend
```

2.1.3 FreeBSD

PowerDNS Authoritative Server is available through the `ports` system:

For the package:

```
# pkg install dns/powerdns
```

To have your system build the port:

```
cd /usr/ports/dns/powerdns/ && make install clean
```

2.1.4 Mac OS X

PowerDNS Authoritative Server is available through Homebrew:

```
$ brew install pdns
```

2.2 After installation

Once installed, *Basic setup: configuring database connectivity* using MySQL or start *migrating* your data.

UPGRADE NOTES

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

Please upgrade to the PowerDNS Authoritative Server 4.0.0 from 3.4.2+. See the [3.X](#) upgrade notes if your version is older than 3.4.2.

3.1 4.1.0 to 4.1.1

- The *Generic MySQL backend* schema has changed: the `notified_serial` column default in the `domains` table has been changed from `INT DEFAULT NULL` to `INT UNSIGNED DEFAULT NULL`:
 - `ALTER TABLE domains MODIFY notified_serial INT UNSIGNED DEFAULT NULL;`

3.2 4.0.X to 4.1.0

- Recursion has been removed, see the *dedicated migration guide*.
- ALIAS record expansion is disabled by default, use *expand-alias* to enable.
- *Your LDAP schema might need to be updated*, because new record types have been added (see below) and the `dnsDomain2` type has been changed.
- The *LDAP Backend* now supports additional Record types
 - NSEC3
 - NSEC3PARAM
 - TLSA
 - CDS
 - CDNSKEY
 - OPENPGPKEY
 - TKEY
 - URI
 - CAA

3.2.1 Changed options

- `experimental-lua-policy-script` option and the feature itself have been completely dropped. We invite you to use [PowerDNS dnsmdist](#) instead.

- As recursion has been removed from the Authoritative Server, the `allow-recursion`, `recursive-cache-ttl` and `recursor` options have been removed as well.
- `default-ksk-algorithms` has been renamed to *default-ksk-algorithm* and only supports a single algorithm name now.
- `default-zsk-algorithms` has been renamed to *default-zsk-algorithm* and only supports a single algorithm name now.

Changed defaults

- The default value of *webserver-allow-from* has been changed from `0.0.0.0, ::/0` to `127.0.0.1, ::1`.

3.2.2 Other changes

The `--with-pgsql`, `--with-pgsql-libs`, `--with-pgsql-includes` and `--with-pgsql-config` configure options have been deprecated. `configure` now attempts to find the PostgreSQL client libraries via `pkg-config`, falling back to detecting `pg_config`. Use `--with-pg-config` to specify a path to a non-default `pg_config` if you have PostgreSQL installed in a non-default location.

The `--enable-libsodium` configure flag has changed from 'no' to 'auto'. This means that if libsodium and its development header are installed, it will be linked in.

The improved *LDAP Backend* backend now requires Kerberos headers to be installed. Specifically, it needs *krb5.h* to be installed.

3.3 4.0.X to 4.0.2

3.3.1 Changed options

Changed defaults

- *any-to-tcp* changed from no to yes

3.4 3.4.X to 4.0.0

3.4.1 Database changes

No changes have been made to the database schema. However, several superfluous queries have been dropped from the SQL backend. Furthermore, the generic SQL backends switched to prepared statements. If you use a non-standard SQL schema, please review the new defaults.

- `insert-ent-query`, `insert-empty-non-terminal-query`, `insert-ent-order-query` have been replaced by one query named `insert-empty-non-terminal-order-query`
- `insert-record-order-query` has been dropped, `insert-record-query` now sets the order-name (or NULL)
- `insert-slave-query` has been dropped, `insert-zone-query` now sets the type of zone

3.4.2 Changed options

Several options have been removed or renamed, for the full overview of all options, see *Authoritative Server Settings*.

Renamed options

The following options have been renamed:

- `experimental-json-interface` ==> *api*
- `experimental-api-readonly` ==> *api-readonly*
- `experimental-api-key` ==> *api-key*
- `experimental-dname-processing` ==> *dname-processing*
- `experimental-dnsupdate` ==> *dnsupdate*
- `allow-dns-update-from` ==> *allow-dnsupdate-from*
- `forward-dnsupdates` ==> *forward-dnsupdate*

Changed defaults

- *default-ksk-algorithm* changed from `rsasha256` to `ecdsa256`
- *default-zsk-algorithm* changed from `rsasha256` to empty

Removed options

The following options are removed:

- `pipebackend-abi-version`, it now a setting per-pipe backend.
- `strict-rfc-axfrs`
- `send-root-referral`

3.4.3 API

The API path has changed to `/api/v1`.

Incompatible change: `SOA-EDIT-API` now follows `SOA-EDIT-DNSUPDATE` instead of `SOA-EDIT` (incl. the fact that it now has a default value of `DEFAULT`). You must update your existing `SOA-EDIT-API` metadata (set `SOA-EDIT` to your previous `SOA-EDIT-API` value, and `SOA-EDIT-API` to `SOA-EDIT` to keep the old behaviour).

3.4.4 Resource Record Changes

Since PowerDNS 4.0.0 the CAA resource record (type 257) is supported. Before PowerDNS 4.0.0 type 257 was used for a proprietary MBOXFW resource record, which was removed from PowerDNS 4.0. Hence, if you used CAA records with 3.4.x (stored in the DB with wrong `type=MBOXFW` but worked fine) and upgrade to 4.0, PowerDNS will fail to parse this records and will throw an exception on all queries for a label with MBOXFW records. Thus, make sure to clean up the records in the DB.

In version 3.X, the PowerDNS Authoritative Server silently ignored records that have a ‘priority’ field (like MX or SRV), but where one was not in the database. In 4.X, *pdnsutil check-zone* will complain about this.

DNS MODES OF OPERATION

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

4.1 Native replication

Native replication is the default, unless other operation is specifically configured. Native replication basically means that PowerDNS will not send out DNS update notifications, nor will react to them. PowerDNS 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 PowerDNS 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 PowerDNS to do so.

Typically, a database slave will be configured as read-only as uni-directional database replication is usually sufficient. A PowerDNS server only requires database write access if it is participating as a master or slave in zone transfers, or has a frontend attached for managing records etc.

4.2 Master operation

When operating as a master, PowerDNS sends out notifications of changes to slaves, which react to these notifications by querying PowerDNS 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](#). Since version 4.0.0, the NOTIFY messages have a TSIG record added (transaction signature) if zone has been configured to use TSIG and feature has been enabled.

Warning: Master support is OFF by default, turn it on by adding *master* to the configuration.

Warning: If you have DNSSEC-signed zones and non-PowerDNS slaves, please check your *SOA-EDIT* settings.

Warning: Notifications are only sent for domains with type MASTER in your backend unless *slave-notify* is enabled.

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, PowerDNS 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:

- `pdns_control notify <domain>` This instructs PowerDNS 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 PowerDNS has trouble figuring out who to notify - which may happen in contrived configurations.

4.3 Slave operation

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

When the freshness of a domain cannot be checked, e.g. because the master is offline, PowerDNS will retry the domain after *slave-cycle-interval* seconds. Every time the domain fails it's freshness check, PowerDNS will hold back on checking the domain for amount of failures * *slave-cycle-interval* seconds, with a maximum of *soa-retry-default* seconds between checks. With default settings, this means that PowerDNS will back off for 1, then 2, then 3 etc. minutes, to a maximum of 60 minutes between checks.

Warning: Slave support is OFF by default, turn it on by adding *slave* to the configuration.

Note: When running PowerDNS via the provided systemd service file, *ProtectSystem* is set to *full*, this means PowerDNS is unable to write to e.g. */etc* and */home*, possibly being unable to write AXFR's zones.

PowerDNS 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. The *retrieve* command is especially useful as it triggers an immediate retrieval of the zone from the configured master.

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.

Since version 4.0.0, PowerDNS requires that masters sign their notifications. During transition and interoperation with other nameservers, you can use options *allow-unsigned-notify* to permit unsigned notifications. For 4.0.0 this is turned on by default, but it might be turned off permanently in future releases.

4.4 Master/Slave Setup Requirements

Generally to enable a Master/Slave setup you have to take care of following properties.

- The *master/slave* state has to be enabled in the respective `/etc/powerdns/pdns.conf` config files.
- The nameservers have to be set up correctly as NS domain records i.e. defining a NS and A record for each slave.
- Master/Slave state has to be configured on a per domain basis in the `domains` table. Namely the `type` column has to be either `MASTER` or `SLAVE` respectively and the slave needs a comma separated list of master node IP addresses in the `master` column in the `domains` table. *more to this topic*.

4.5 IXFR: incremental zone transfers

If the 'IXFR' zone metadata item is set to 1 for a zone, PowerDNS will attempt to retrieve zone updates via IXFR.

Warning: If a slave zone changes from non-DNSSEC to DNSSEC, an IXFR update will not set the PRE-SIGNED flag. In addition, a change in NSEC3 mode will also not be picked up.

In such cases, make sure to delete the zone contents to force a fresh retrieval.

Finally, IXFR updates that “plug” Empty Non Terminals do not yet remove ENT records. A ‘`pdnsutil rectify-zone`’ may be required.

PowerDNS itself is currently only able to retrieve updates via IXFR. It can not serve IXFR updates.

4.6 Supermaster: automatic provisioning of slaves

PowerDNS 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 PowerDNS already knows that it is a slave for a domain.

However, a notification from a supermaster carries more persuasion. When PowerDNS determines that a notification comes from a supermaster and it is bonafide, it 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
- If your master sends signed NOTIFY it will mark that TSIG key as the TSIG key used for retrieval as well
- If you turn off *allow-unsigned-supermaster*, then your supermaster(s) are required to sign their notifications.

Warning: If you use another PowerDNS server as master and have DNSSEC enabled on that server please don’t forget to rectify the domains after every change. If you don’t do this there is no SOA record available and one requirement will fail.

So, to benefit from this feature, a backend needs to know about the IP address of the supermaster, and how PowerDNS 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.

Note: Removal of zones provisioned using the supermaster must be done on the slaves themselves. As there is no way to signal this removal from the master to the slave.

4.7 Modifying a slave zone using a script

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.

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");
```

Warning: 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 the table is empty, no record is added). If you want your record(s) to be appended after the matching record, return 1 and table of record(s). 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, record)

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

  -- Grab each _tstamp TXT record and add a time stamp
  if record:qtype() == pdns.TXT and string.starts(record:qname:toString(), "_  
→tstamp.") then
    resp = {}
    resp[1] = {
      qname    = record:qname():toString(),
      qtype    = record:qtype(),
```

(continues on next page)

(continued from previous page)

```
        ttl      = record:ttl(),
        content = os.date("Ver %Y%m%d-%H:%M")
    }
    return 0, resp
end

-- Append A records with this TXT
if record:qtype() == pdns.A then
    resp = {}
    resp[1] = {
        qname    = record:qname:toString(),
        qtype    = pdns.TXT,
        ttl      = 99,
        content   = "Hello Ahu, again!"
    }
    return 1, 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. A records are appended with a TXT record. All other records are unhandled.

MIGRATING TO POWERDNS

Before migrating to PowerDNS a few things should be considered.

PowerDNS does not operate as a *Slave operation* or *Master operation* server with all backends. The *Generic SQL* and *BIND* backends have the ability to act as master or slave. See the *table of backends* which other backends support these modes.

5.1 Using AXFR to a Slave-Capable Backend

The easiest way to migrate all your zones from your old infrastructure to PowerDNS is to add all your domains as a slave domain with your current master as the master, wait for the zones to be transferred and change the zones to master. Make sure *slave* is set to “yes” in your `pdns.conf`.

5.1.1 To A Generic SQL Backend

Note: This assumes the schema provided with PowerDNS is in place

In order to migrate to a Generic SQL backend, add all your domains to the ‘domains’ table with the IP of your current master. On your current master, make sure that this master allows AXFRs to this new slave.

```
INSERT INTO domains (name,type,master) VALUES ('example.net', 'SLAVE', '198.51.100.
↪101');
```

Then start PowerDNS and wait for all the zones to be transferred. If this server is the new *master*, change the type of domain in the database:

```
UPDATE domains set type='MASTER' where type='SLAVE';
```

And set *master* to “yes” in your `pdns.conf` and restart PowerDNS.

Or, if you want to use *native*:

```
UPDATE domains set type='NATIVE' where type='SLAVE';
```

5.1.2 To the BIND backend

Create a `named.conf` with all the domains as slave domains, e.g.:

```
zone "example.net" in {
    type slave;
    file "/var/lib/powerdns/zones/example.net.zone";
    masters {
```

(continues on next page)

(continued from previous page)

```
198.51.100.101;
};
};
```

Make sure the directory is writable for the `pdns_server` process and that `bind-config` parameter references this file. Now start PowerDNS and wait until all zones are transferred. Now you can change the zone type to master:

```
zone "example.net" in {
    type master;
    file "/var/lib/powerdns/zones/example.net.zone";
};
```

Don't forget to enable `master` in your `pdns.conf` and restart, or if this setting was already set, use `pdns_control rediscover` to load these zones as master zones.

5.2 From zonefiles to PowerDNS

5.2.1 Using the BIND backend

To use the bind backend, set `launch=bind` and `bind-config=/path/to/named.conf` in your `pdns.conf`. Note that PowerDNS will not honor any options from `named.conf`, it will only use the zone statements. See the [Bind backend](#) documentation for more information.

5.2.2 To a Generic SQL backend

There are several methods to migrate to a *Generic SQL* backend.

Using zone2sql

To migrate, the `zone2sql` tool is provided. This tool parses a BIND `named.conf` file and zone files and outputs SQL on standard out, which can then be fed to your database. It understands the Bind master file extension `$GENERATE` and will also honour `$ORIGIN` and `$TTL`.

For backends supporting slave operation, there is also an option to keep slave zones as slaves, and not convert them to native operation.

`zone2sql` can generate SQL for nearly all the Generic SQL backends. See its manpage <manpages/zone2sql.1> for more information.

An example call to `zone2sql` could be:

```
zone2sql --named-conf=/path/to/named.conf --gmysql | mysql -u pdns -p pdns-db
```

This will generate the SQL statements for the *Generic MySQL* and pipe them into the `pdns-db` database in MySQL.

Using pdnsutil load-zone

The `pdnsutil` tool has a `load-zone` command that ingests a zone file and imports it into the first backend that is capable of hosting it.

To import, configure the backend and run `pdnsutil load-zone example.com /tmp/example.com.com.zone` to import the `example.com` domain from the `/tmp/example.com.zone` file. The zone is imported atomically (i.e. it is fully imported, or not) and any existing records for that zone are overwritten.

5.3 Migrating Data from one Backend to Another Backend

Note: This is experimental feature.

Syntax: `pdnsutil b2b-migrate OLD NEW`

This tool lets you migrate data from one backend to another, it moves all data, including zones, metadata and crypto keys (if present). Some example use cases are moving from Bind style zonefiles to SQL based, or other way around, or moving from MyDNS to gMySQL.

5.3.1 Prerequisites

- Target backend must support same features as source from set of domains, zones, metadata, DNSSEC and TSIG. See *Backend Capabilities*
- There must be no data in the target backend, otherwise the migration will fail. This is checked.

You can perform live upgrade with this tool, provided you follow the procedure.

5.3.2 Moving from source to target

- Take backups of everything.
- Configure both backends to `pdns.conf`, if you have source configured, you can just add target backend. **DO NOT RESTART AUTH SERVER BEFORE YOU HAVE FINISHED**
- Then run `pdnsutil b2b-migrate old new`, the old and new being configuration prefixes in `pdns.conf`. If something goes wrong, make sure you properly clear **ALL** data from target backend before retrying.
- Remove (or comment out) old backend from `pdns.conf`, and run `pdnsutil rectify-all-zones` and `pdnsutil check-all-zones` to make sure everything is OK.
- If everything is OK, then go ahead to restart your PowerDNS service. Check logs to make sure everything went ok.

RUNNING AND OPERATING

PowerDNS is normally controlled via a SysV-style `init.d` script, often located in `/etc/init.d` or `/etc/rc.d/init.d`. For Linux distributions with `systemd`, a service file is provided (either in the package or in the `contrib` directory of the tarball).

Furthermore, PowerDNS can be run on the foreground for testing or in other `init`-systems that supervise processes.

6.1 Guardian

When the `init`-system of the Operating System does not properly supervise processes, like SysV `init`, it is recommended to run PowerDNS with the *guardian* option set to 'yes'.

When launched with `guardian=yes`, `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. Requests that require data from the actual nameserver are passed to the inner process as well.

6.2 Logging to syslog on systemd-based operating systems

By default, logging to `syslog` is disabled in the `systemd` unit file to prevent the service logging twice, as the `systemd` journal picks up the output from the process itself.

Removing the `--disable-syslog` option from the `ExecStart` line using `systemctl edit --full pdns` enables logging to `syslog`.

6.3 Logging to syslog

This chapter assumes familiarity with `syslog`, the unix logging device. PowerDNS logs messages with different levels. The more urgent the message, the lower the 'priority'.

By default, PowerDNS 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 PowerDNS 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 PowerDNS 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 PowerDNS 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`

6.4 Controlling A Running PowerDNS Server

As a DNS server is critical infrastructure, downtimes should be avoided as much as possible. Even though PowerDNS (re)starts very fast, it offers a way to control it while running.

6.4.1 Control Socket

The controlsocket is the means to contact a running PowerDNS process. Over this socket, instructions can be sent using the `pdns_control` program. The control socket is called `pdns.controlsocket` and is created inside the *socket-dir*.

6.4.2 pdns_control

To communicate with PowerDNS Authoritative Server over the controlsocket, the `pdns_control` command is used. 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 *Master operation*.

For all supported `pdns_control` commands and options, see *the manpage* and the output of `pdns_control --help` on your system.

6.5 The SysV init script

This script supplied with the PowerDNS source accepts the following commands:

- **monitor:** Monitor is a special way to view the daemon. It executes PowerDNS in the foreground with a lot of logging turned on, which helps in determining startup problems. Besides running in the foreground, the raw PowerDNS 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 PowerDNS 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 PowerDNS instances.
- **restart:** Restarts PowerDNS if it was running, starts it otherwise.
- **status:** Query PowerDNS 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 PowerDNS process.
- **stop:** Requests that PowerDNS stop. Again, does not confirm success. Success can be ascertained with the `status` command.

- `dump`: Dumps a lot of statistics of a running PowerDNS 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`: Dump statistics in mrtg format. See the performance [Counters](#) documentation.

Note: Packages provided by Operating System vendors might support different or less commands.

6.6 Running in the foreground

One can run PowerDNS in the foreground by invoking the `pdns_server` executable. Without any options, it will load the `pdns.conf` and run. To make sure PowerDNS starts in the foreground, add the `--daemon=no` option.

All [settings](#) can be added on the commandline. e.g. to test a new database config, you could start PowerDNS like this:

```
pdns_server --no-config --daemon=no --local-port=5300 --launch=gmysql --gmysql-  
↪user=my_user --gmysql-password=mypassword
```

This starts PowerDNS without loading on-disk config, in the foreground, on all network interfaces on port 5300 and starting the *gmysql* backend.

SECURITY OF POWERDNS

PowerDNS has several options to easily allow it to run more securely. Most notable are the [chroot](#), [setuid](#) and [setgid](#) options.

For Security Advisories, see the [dedicated page](#).

7.1 PowerDNS Security Policy

If you have a security problem to report, please email us at both security@powerdns.com 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 this documentation.

As of the 9th of September 2016, no actual security problems with PowerDNS Authoritative Server 3.4.10, Recursor 3.6.3, Recursor 3.7.2, 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](#).

7.1.1 HackerOne

Security issues can also be reported on our [HackerOne page](#) and might fetch a bounty. Do note that only the PowerDNS software is in scope for the HackerOne program, not our websites or other infrastructure.

7.1.2 Disclosure Policy

- Let us know as soon as possible upon discovery of a potential security issue, and we'll make every effort to quickly resolve the issue.
- Provide us a reasonable amount of time to resolve the issue before any disclosure to the public or a third-party.
- We will always credit researchers in our [Security Advisories](#).

For additional information on PowerDNS security, PowerDNS security incidents and PowerDNS security policy, see [PowerDNS Security Policy](#).

7.2 Securing the Process

7.2.1 Running as a less privileged identity

By specifying `setuid` and `setgid`, PowerDNS 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 PowerDNS as the user ‘nobody’ is in fact quite powerful on most systems.

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

7.2.2 Jailing the process in a chroot

The `chroot` option secures PowerDNS 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 The PowerDNS, 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 PowerDNS 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 PowerDNS.

7.3 Security Considerations

In general, make sure that the PowerDNS 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 PowerDNS! 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.

7.4 Security Polling

PowerDNS products can poll the security status of their respective versions. This polling, naturally, happens over DNS. If the result is that a given version has a security problem, the software will report this at level ‘Error’ during startup, and repeatedly during operations.

By default, security polling happens on the domain ‘secpoll.powerdns.com’, but this can be changed with the `security-poll-suffix`. If this setting is made empty, no polling will take place. Organizations wanting to host their own security zones can do so by changing this setting to a domain name under their control.

To make this easier, the zone used to host secpoll.powerdns.com is available [here](#).

To enable distributors of PowerDNS to signal that they have backported versions, the `PACKAGEVERSION` compilation-time macro can be used to set a distributor suffix.

7.4.1 Details

PowerDNS software sadly sometimes has critical security bugs. Even though we send out notifications of these via all channels available, we find that not everybody actually find out about our security releases.

To solve this, PowerDNS software will start polling for security notifications, and log these periodically. Secondly, the security status of the software will be reported using the built-in metrics. This allows operators to poll for the PowerDNS security status and alert on it.

In the implementation of this idea, we have taken the unique role of operating system distributors into account. Specifically, we can deal with backported security fixes.

Finally, this feature can be disabled, or operators can have the automated queries point at their own status service.

Implementation

PowerDNS software periodically tries to resolve `'auth-x.y.z.security-status.secpoll.powerdns.com|TXT'` or `'recursor-x.y.z.security-status.secpoll.powerdns.com'`.

The data returned is in one of the following forms:

- NXDOMAIN or resolution failure -> 0
- "1 Ok" -> 1
- "2 Upgrade recommended for security reasons, see ..." -> 2
- "3 Upgrade mandatory for security reasons, see ..." -> 3

In cases 2 or 3, periodic logging commences. The metric security-status is set to 2 or 3 respectively. If at a later date, resolution fails, the security-status is not reset to 1. It could be lowered however if we discover the security status is less urgent than we thought.

If resolution fails, and the previous security-status was 1, the new security-status becomes 0 ('no data'). If the security-status was higher than 1, it will remain that way, and not get set to 0.

In this way, security-status of 0 really means 'no data', and can not mask a known problem.

Distributions

Distributions frequently backport security fixes to the PowerDNS versions they ship. This might lead to a version number that is known to us to be insecure to be secure in reality.

To solve this issue, PowerDNS can be compiled with a distribution setting which will move the security polls from: `'auth-x.y.z.security-status.secpoll.powerdns.com'` to `'auth-x.y.z-n.debian.security-status.secpoll.powerdns.com'`.

Note two things, one, there is a separate namespace for debian, and secondly, we use the package version of this release. This allows us to know that 4.0.1-1 (say) is insecure, but that 4.0.1-2 is not.

Configuration Details

The configuration setting `security-poll-suffix` is by default set to `'secpoll.powerdns.com'`. If empty, nothing is polled. This can be moved to `'secpoll.yourorganization.com'`.

If compiled with `PACKAGEVERSION=3.1.6-abcde.debian`, queries will be sent to `"auth-3.1.6-abcde.debian.security-status.security-poll-suffix"`.

Delegation

If a distribution wants to host its own file with version information, we can delegate `dist.security-status.secpoll.powerdns.com` to their nameservers directly.

PERFORMANCE AND TUNING

In general, best performance is achieved on recent Linux 4.x kernels and using MySQL, although many of the largest PowerDNS installations are based on PostgreSQL. FreeBSD also performs very well.

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

Warning: When deploying (large scale) IPv6, please be aware some Linux distributions leave IPv6 routing cache tables at very small default values. Please check and if necessary raise `sysctl net.ipv6.route.max_size`.

8.1 Performance related settings

When PowerDNS starts up it creates a number of threads to listen for packets. This is configurable with the *receiver-threads* setting which defines how many sockets will be opened by the powerdns process. In versions of linux before kernel 3.9 having too many receiver threads set up resulted in decreased performance due to socket contention between multiple CPUs - the typical sweet spot was 3 or 4. For optimal performance on kernel 3.9 and following with *reuseport* enabled you'll typically want a receiver thread for each core on your box if backend latency/performance is not an issue and you want top performance.

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 which says how many distributors will be opened for each receiver thread. Of special importance is the choice between 1 or more backends. In case of only 1 thread, PowerDNS reverts to unthreaded operation which may be a lot faster, depending on your operating system and architecture.

Other very important settings are *cache-ttl*. PowerDNS 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. Please be aware that if any TTL in the answer is shorter than this setting, the packet cache will respect the answer's shortest TTL.

Some PowerDNS operators set cache-ttl to many hours or even days, and use *pdns_control purge* to selectively or globally notify PowerDNS 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 PowerDNS 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.

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* off.

8.2 Packet Cache

PowerDNS 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 20 seconds and can be changed by setting `cache-ttl`. 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 packet cache. If your backend is memory based and does not lead to context switches, the packet cache may actually hurt performance.

Changed in version 4.1.0: The maximum size of the packet cache is controlled by the *max-packet-cache-entries* entries. Before that both the query cache and the packet cache used the *max-cache-entries* setting.

8.3 Query Cache

Besides entire packets, PowerDNS 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’, PowerDNS must first check for a CNAME for ‘www.powerdns.com’.

The Query Cache caches these backend queries, many of which are quite repetitive. The maximum number of entries in the cache is controlled by the `max-cache-entries` setting. Before 4.1 this setting also controls the maximum number of entries in the packet cache.

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.

8.4 Performance Monitoring

A number of counters and variables are set during PowerDNS Authoritative Server operation.

8.4.1 Counters

All counters that show the “number of X” count since the last startup of the daemon.

corrupt-packets

Number of corrupt packets received

deferred-cache-inserts

Number of cache inserts that were deferred because of maintenance

deferred-cache-lookup

Number of cache lookups that were deferred because of maintenance

deferred-packetcache-inserts

Number of packet cache inserts that were deferred because of maintenance

deferred-packetcache-lookup

Number of packet cache lookups that were deferred because of maintenance

dnsupdate-answers

Number of DNS update packets successfully answered

dnsupdate-changes

Total number of changes to records from DNS update

dnsupdate-queries

Number of DNS update packets received

dnsupdate-refused

Number of DNS update packets that were refused

incoming-notifications

Number of NOTIFY packets that were received

key-cache-size

Number of entries in the key cache

latency

Average number of microseconds a packet spends within PowerDNS

meta-cache-size

Number of entries in the metadata cache

overload-drops

Number of questions dropped because backends overloaded

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-q

Number of packets waiting for database attention

query-cache-hit

Number of hits on the *Query Cache*

query-cache-miss

Number of misses on the *Query Cache*

query-cache-size

Number of entries in the query cache

rd-queries

Number of packets sent by clients requesting recursion (regardless of if we'll be providing them with recursion).

recursing-answers

Number of packets we supplied an answer to after recursive processing

recursing-questions

Number of packets we performed recursive processing for.

recursion-unanswered

Number of packets we sent to our recursor, but did not get a timely answer for.

security-status

Security status based on *Security Polling*.

servfail-packets

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

signature-cache-size

Number of entries in the signature cache

signatures

Number of DNSSEC signatures created

sys-msec

Number of CPU milliseconds sent in system time

tcp-answers-bytes

Total number of answer bytes sent over TCP

tcp-answers

Number of answers sent out over TCP

tcp-queries

Number of questions received over TCP

tcp4-answers-bytes

Total number of answer bytes sent over TCPv4

tcp4-answers

Number of answers sent out over TCPv4

tcp4-queries

Number of questions received over TCPv4

tcp6-answers-bytes

Total number of answer bytes sent over TCPv6

tcp6-answers

Number of answers sent out over TCPv6

tcp6-queries

Number of questions received over TCPv6

timedout-packets

Amount of packets that were dropped because they had to wait too long internally

udp-answers-bytes

Total number of answer bytes sent over UDP

udp-answers

Number of answers sent out over UDP

udp-do-queries

Number of queries received with the DO (DNSSEC OK) bit set

udp-in-errors

Number of packets, received faster than the OS could process them

udp-noport-errors

Number of UDP packets where an ICMP response was received that the remote port was not listening

udp-queries

Number of questions received over UDP

udp-recvbuf-errors

Number of errors caused in the UDP receive buffer

udp-sndbuf-errors

Number of errors caused in the UDP send buffer

udp4-answers-bytes

Total number of answer bytes sent over UDPv4

udp4-answers

Number of answers sent out over UDPv4

udp4-queries

Number of questions received over UDPv4

udp6-answers-bytes

Total number of answer bytes sent over UDPv6

udp6-answers

Number of answers sent out over UDPv6

udp6-queries

Number of questions received over UDPv6

uptime

Uptime in seconds of the daemon

user-msec

Number of milliseconds spend in CPU 'user' time

8.4.2 Ring buffers

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

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

The following ringbuffers are available:

- **logmessages**: All messages logged
- **noerror-queries**: Queries for existing records but for a type we don't have. 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 AAAA query for pdns.powerdns.com looks like pdns.powerdns.com/AAAA.
- **nxdomain-queries**: Queries for non-existing records within existing domains. If PowerDNS 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**: All UDP queries seen.
- **remotes**: Remote server IP addresses. Number of hosts querying PowerDNS. Be aware that UDP is anonymous - person A can send queries that appear to be coming from person B.
- **remote-corrupts**: Remotes sending corrupt packets. Hosts sending PowerDNS 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.
- **remote-unauth**: Remotes querying domains for which we are not authoritative. It may happen that there are misconfigured hosts on the internet which are configured to think that a PowerDNS installation is in fact a resolving nameserver. These hosts will not get useful answers from PowerDNS. This buffer lists hosts sending queries for domains which PowerDNS does not know about.
- **servfail-queries**: Queries that could not be answered due to backend errors. 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, PowerDNS sends out a 'servfail' packet indicating that it was unable to answer the question. This buffer shows which queries have been causing servfails.

- **unauth-queries:** Queries for domains that we are not authoritative for. If a domain is delegated to a PowerDNS 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.

8.4.3 Sending metrics to Graphite/Metronome over Carbon

For carbon/graphite/metronome, we use the following namespace. Everything starts with 'pdns.', which is then followed by the local hostname. Thirdly, we add 'auth' to signify the daemon generating the metrics. This is then rounded off with the actual name of the metric. As an example: 'pdns.ns1.auth.questions'.

Care has been taken to make the sending of statistics as unobtrusive as possible, the daemons will not be hindered by an unreachable carbon server, timeouts or connection refused situations.

To benefit from our carbon/graphite support, either install Graphite, or use our own lightweight statistics daemon, Metronome, currently available on [GitHub](#).

To enable sending metrics, set *carbon-server*, possibly *carbon-interval* and possibly *carbon-ourname* in the configuration.

Warning: If your hostname includes dots, they will be replaced by underscores so as not to confuse the namespace.

If you include dots in *carbon-ourname*, they will **not** be replaced by underscores. As PowerDNS assumes you know what you are doing if you override your hostname.

DNSSEC

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

In PowerDNS, 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 key records, signatures and (hashed) denials of existence automatically.

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

```
$ pdnsutil secure-zone powerdnssec.org
```

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

9.1 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 PowerDNS. 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, ECDSA or GOST), the public part is published in 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 PowerDNS zone can either be operated in NSEC or in one of two NSEC3 modes (‘inclusive’ and ‘narrow’).

9.2 DNSSEC Profile and Support

PowerDNS 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 PowerDNS 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.

9.2.1 Supported Algorithms

Supported Algorithms (See the [IANA website](#) for more information):

- RSASHA1 (algorithm 5, algorithm 7)
- RSASHA256 (algorithm 8)
- RSASHA512 (algorithm 10)
- ECC-GOST (algorithm 12)
- ECDSA (algorithm 13 and 14)
- ed25519 (algorithm 15)
- ed448 (algorithm 16)

For the DS records, these [digest types](#) are supported:

- SHA-1 (algorithm 1)
- SHA-256 (algorithm 2)
- GOST R 34.11-94 (algorithm 3)
- SHA-384 (algorithm 4)

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
- **RFC 6605**: Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC
- **RFC 8080**: Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC

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

When using OpenSSL for ECDSA signatures (this is default), starting from OpenSSL 1.1.0, the algorithm used is resilient against PRNG failure, while not strictly conforming to [RFC 6979](#).

Note: Actual supported algorithms depend on the crypto-libraries PowerDNS was compiled against. To check the supported DNSSEC algorithms in your build of PowerDNS, run `pdnsutil list-algorithms`.

9.3 DNSSEC Modes of Operation

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 *Security of PowerDNS* and *Performance and Tuning*.

9.3.1 Online Signing

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.

Records, Keys, signatures, hashes within PowerDNS in online signing mode

Within PowerDNS 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 inactive.

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.

(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.

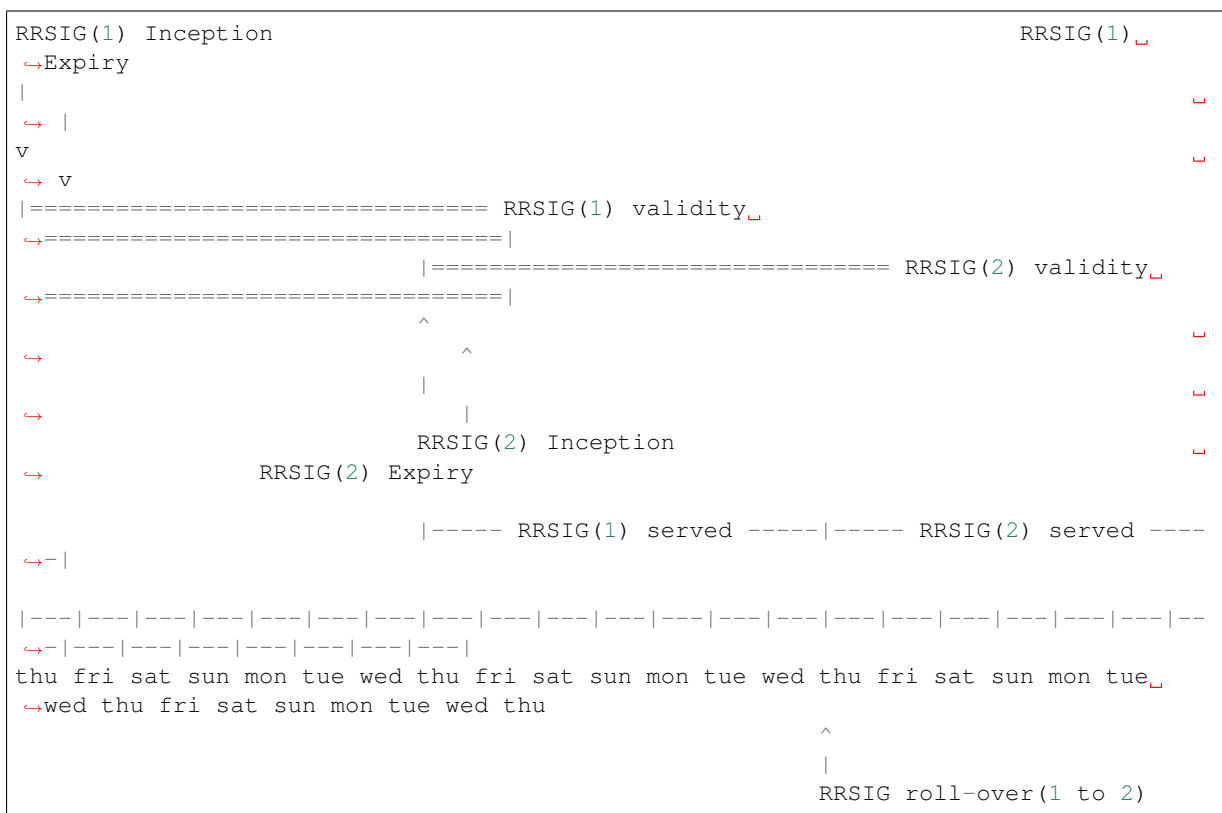
NSEC3 in ‘narrow’ mode uses additional hashing calculations to provide hashed secure denial of existence ‘on the fly’, without further involving the database.

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 this period is 3 weeks. This period starts at most a week in the past, and continues at least a week into the future. This interval jumps with one-week increments every Thursday.

The time period used is always calculated based on the moment of rollover. The inception timestamp is the most recent Thursday 00:00:00 UTC, which is exactly one week ago at the moment of rollover. The expiry timestamp is the Thursday 00:00:00 UTC two weeks later from the moment of rollover. Graphically, it looks like this:



At all times, only one RRSIG per signed RRset per ZSK is served when responding to clients.

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’.

PowerDNS also serves the DNSKEY records in live-signing mode. Their TTL is derived from the SOA records *minimum* field. When using NSEC3, the TTL of the NSEC3PARAM record is also derived from that field.

9.3.2 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*.

Even in this mode, PowerDNS will synthesize NSEC(3) records itself because of its architecture. RRSIGs of these NSEC(3) will still need to be imported. See the [Presigned migration guide](#).

9.3.3 Front-signing

As a special feature, PowerDNS 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 `pdnsutil rectify-zone` is run to make sure that all DNSSEC-related fields are set correctly in the backend.

9.3.4 Signed AXFR

An outgoing zone transfer from a signing master contains all information required for the receiving party to rectify the zone without knowing the keys, such as signed NSEC3 records for empty non-terminals. The zone is not required to be rectified on the master.

Signatures and Hashing is similar as described in [Online Signing](#).

9.3.5 BIND-mode operation

The *bindbackend* can manage keys in an SQLite3 database without launching a separate *gsqli3* backend.

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

Note: This *sqlite* database is different from the database used for the regular *SQLite 3 backend*.

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

9.3.6 Hybrid BIND-mode operation

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. See the [backend specific documentation](#) on how to initialize the database and backend.

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');
```

The *SQLite 3 backend* probably complements BIND mode best, since it does not require a database server process.

Note: The `sqlite3` database must be created using the normal schema for this backend. The database created with `pdnsutil create-bind-db` will not work in this backend.

9.4 `pdnsutil` and DNSSEC

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

For a list of available commands, see the *manpage*.

9.4.1 DNSSEC Defaults

Since version 4.0, when securing a zone using `pdnsutil secure-zone`, a single ECDSA (algorithm 13, ECDSAP256SHA256) key is generated that is used as ZSK. Before 4.0, 3 RSA (algorithm 8) keys were generated, one as the KSK and two ZSKs. As all keys are online in the database, it made no sense to have this split-key setup.

The default negative answer strategy is NSEC.

Note: Not all registrars support algorithm 13.

9.5 Migrating (Signed) Zones to PowerDNS

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

9.5.1 From an existing PowerDNS installation

To migrate an existing database-backed PowerDNS installation, ensure you are running at least PowerDNS 3.3.3 and preferably 3.4 or newer.

If you run an older version of PowerDNS, please upgrade to 3.4 and apply all the changes in database schemas as shown in the *upgrade documentation*.

Warning: Once the relevant `backend-dnssec` switch has been set, stricter rules apply for filling out the database! The short version is: run `pdnsutil rectify-all-zones`, even those not secured with DNSSEC! For more information, see the *Handling DNSSEC signed zones*.

To deliver a correctly signed zone with the *DNSSEC Defaults*, invoke:

```
pdnsutil secure-zone ZONE
```

To view the DS records for this zone (to transfer to the parent zone), run

```
pdnsutil show-zone ZONE
```

For a more traditional setup with a KSK and a ZSK, use the following sequence of commands:

```
pdnsutil add-zone-key ZONE ksk 2048 active rsasha256
pdnsutil add-zone-key ZONE zsk 1024 active rsasha256
pdnsutil add-zone-key ZONE zsk 1024 inactive rsasha256
```

This will add a 2048-bit RSA Key Signing Key and two 1024-bit RSA Zone Signing Keys. One of the ZSKs is inactive and can be rolled to if needed.

9.5.2 From existing non-DNSSEC, non-PowerDNS setups

It is recommended to *migrate to PowerDNS* before securing your zones. After that, see the instructions *above*.

9.5.3 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, ldns-sign or dnssec-sign.

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

If you import presigned zones into your database, please do not import the NSEC or NSEC3 records. PowerDNS will synthesize these itself. Putting them in the database might cause duplicate records in responses. *zone2sql* filters NSEC and NSEC3 automatically.

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

9.5.4 From existing DNSSEC non-PowerDNS setups, live signing

The `pdnsutil` tool features the option to import zone keys in the industry standard private key format, version 1.2. To import an existing KSK, use

```
pdnsutil import-zone-key ZONE 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!

9.5.5 Secure transfers

PowerDNS supports secure DNSSEC transfers as described in [draft-koch-dnsop-dnssec-operator-change](#). If the *direct-dnskey* option is enabled the foreign DNSKEY records stored in the database are added to the keyset and signed with the KSK. Without the *direct-dnskey* option DNSKEY records in the database are silently ignored.

9.6 Operational instructions

Several How to's describe operational practices with DNSSEC:

- [KSK Rollover](#)

- [KSK Rollover using CDS & CDNSKEY Key Rollover](#)
- [ZSK Rollover](#)

Below, frequently used commands are described:

9.6.1 Publishing a DS

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

9.6.2 Going insecure

```
pdnsutil disable-dnssec ZONE
```

Warning: Going insecure with a zone that has a DS record in the parent zone will make the zone BOGUS. Make sure the parent zone removes the DS record *before* going insecure.

9.6.3 Setting the NSEC modes and parameters

As stated earlier, PowerDNS uses NSEC by default. If you want to use NSEC3 instead, issue:

```
pdnsutil set-nsec3 ZONE [PARAMETERS]
```

e.g.

```
pdnsutil set-nsec3 example.net '1 0 1 ab'
```

The quoted part is the content of the NSEC3PARAM records, as defined in [5155](#), in order:

- Hash algorithm, should always be 1 (SHA1)
- Flags, set to 1 for **NSEC3 Opt-out**, this best set as 0
- Number of iterations of the hash function, read [RFC 5155, Section 10.3](#) for recommendations
- Salt to apply during hashing, in hexadecimal, or – to use no salt

To convert a zone from NSEC3 to NSEC operations, run:

```
pdnsutil unset-nsec3 ZONE
```

Warning: Don't change from NSEC to NSEC3 (or the other way around) for zones with algorithm 5 (RSASHA1), 6 (DSA-NSEC3-SHA1) or 7 (RSASHA1-NSEC3-SHA1).

9.6.4 SOA-EDIT: ensure signature freshness on slaves

As RRSIGs can expire, slave servers need to know when to re-transfer the zone. In most implementations (BIND, NSD), this is done by re-signing the full zone outside of the nameserver, increasing the SOA serial and serving the new zone on the master.

With PowerDNS in Live-signing mode, the SOA serial is not increased by default when the RRSIG dates are rolled.

For zones that use [Native replication](#) replication PowerDNS will serve valid RRSIGs on all servers.

For *master* zones (where replication happens by means of AXFR), PowerDNS slaves will automatically re-transfer the zone when it notices the RRSIGs have changed, even when the SOA serial is not increased. This ensures the zone never serves old signatures.

If your DNS setup uses non-PowerDNS slaves, the slaves need to know when the signatures have been updated. This can be accomplished by setting the *SOA-EDIT* metadata for DNSSEC signed zones. This value controls how the value of the SOA serial is modified by PowerDNS.

Note: The SOA serial in the datastore will be untouched, SOA-EDIT is applied to DNS answers with the SOA record.

The *default-soa-edit* or *default-soa-edit-signed* configuration options can instead be set to ensure SOA-EDIT is set for every zone.

Possible SOA-EDIT values

The ‘inception’ refers to the time the RRSIGs got updated in *live-signing mode*. This happens every week (see *Signatures*). The inception time does not depend on local timezone, but some modes below will use localtime for representation.

INCREMENT-WEEKS

Increments the serial with the number of weeks since the UNIX epoch. This should work in every setup; but the result won’t look like YYYYMMDDSS anymore.

For example: a serial of 12345678 will become 12348079 on Wednesday 13th of January 2016 (2401 weeks after the epoch).

INCEPTION-EPOCH

Sets the new SOA serial number to the maximum of the old SOA serial number, and age in seconds of the last inception. This requires your backend zone to use the number of seconds since the UNIX epoch as SOA serial. The result is still the age in seconds of the last change to the zone, either by operator changes to the zone or the ‘addition’ of new RRSIGs.

As an example, a serial of 12345678 becomes 1452124800 on Wednesday 13th of January 2016.

INCEPTION-INCREMENT

Uses YYYYMMDDSS format for SOA serial numbers. If the SOA serial from the backend is within two days after inception, it gets incremented by two (the backend should keep SS below 98). Otherwise it uses the maximum of the backend SOA serial number and inception time in YYYYMMDD01 format. This requires your backend zone to use YYYYMMDDSS as SOA serial format. Uses localtime to find the day for inception time.

This changes a serial of 2015120810 to 2016010701 on Wednesday 13th of January 2016.

INCEPTION (not recommended)

Deprecated since version 4.1.0: Removed in this release

Sets the SOA serial to the last inception time in YYYYMMDD01 format. Uses localtime to find the day for inception time.

Warning: The SOA serial will only change on inception day, so changes to the zone will get visible on slaves only on the following inception day.

INCEPTION-WEEK (not recommended)

Deprecated since version 4.1.0: Removed in this release

Sets the SOA serial to the number of weeks since the epoch, which is the last inception time in weeks.

Warning: Same problem as INCEPTION.

EPOCH

Sets the SOA serial to the number of seconds since the epoch.

Warning: Don't combine this with AXFR - the slaves would keep refreshing all the time. If you need fast updates, sync the backend databases directly with incremental updates (or use the same database server on the slaves)

NONE

Ignore *default-soa-edit* and/or *default-soa-edit-signed* settings.

9.6.5 Security

During typical PowerDNS operation, the private part of the signing keys are 'online', which can be compared to operating an HTTPS server, where the private key 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.

9.6.6 Performance

DNSSEC has a performance impact, mostly measured in terms of additional memory used for the signature caches. In addition, on startup or AXFR-serving, a lot of signing needs to happen.

Most best practices are documented in [RFC 6781](#).

9.7 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 `pdnsutil secure-zone`.

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.

Depending on your master/slave setup, you may need to tinker with the *SOA-EDIT* metadata on your master. This is described in the *SOA-EDIT: ensure signature freshness on slaves* section.

9.7.1 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:

- Larger than 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.

9.8 PKCS#11 support

Note: This feature is experimental, use at your own risk!

Deprecated since version 4.0.0: slot IDs are deprecated, and you are expected to use slot label instead

To enable it, compile PowerDNS Authoritative Server using `--enable-experimental-pkcs11` flag on configure. This requires you to have p11-kit libraries and headers.

You can also log on to the tokens after starting server, in this case you need to edit your PKCS#11 cryptokey record and remove PIN or set it empty. PIN is required for assigning keys to zone.

9.8.1 Using with SoftHSM

Warning: Due to an interaction between [SoftHSM](#) and [Botan](#), the PowerDNS Authoritative Server **will most likely** crash on exit when built with `--enable-botan1.10 --enable-experimental-pkcs11`. This is the case with the packages provided from the PowerDNS repositories.

To test this feature, a software HSM can be used. It is **not recommended** to use this in production.

Instructions on how to setup SoftHSM to work with the feature after compilation on ubuntu/debian (tested with Ubuntu 12 and 14). - `apt-get install softhsm p11-kit opensc` - create directory `/etc/pkcs11/modules` - Add file called ‘softhsm’ there with (on newer versions, use `softhsm.module`)
module: `/home/cmouse/softhsm/lib/softhsm/libsofthsm.so` managed: `yes` - Verify it works: `p11-kit -l` - Create at least two tokens (sk and zsk) with (slot-number starts from 0)

```
...
sudo softhsm --init-token --slot slot-number --label zone-ksk|zone-zsk --pin some-
↪pin --so-pin another-pin
...
```

- Using pkcs11-tool, initialize your new keys.

```
sudo pkcs11-tool --module=/home/cmouse/softhsm/lib/softhsm/libsofthsm.so -l -p some-pin -k --key-type RSA:2048 -a zone-ksk|zone-zsk --slot-index slot-number
```

- Assign the keys using (note that token label is not necessarily same as object label, see p11-kit -l)

```
pdnsutil hsm assign zone rsasha256 ksk|zsk softhsm token-label pin zone-ksk|zsk
```

- Verify that everything worked, you should see valid data there

```
pdnsutil show-zone zone
```

- SoftHSM signatures are fast enough to be used in live environment.

9.8.2 Using CryptAS

Instructions on how to use CryptAS `Athena IDProtect Key USB Token V2J` <<http://www.cryptoshop.com/products/smartcards/idprotect-key-j-laser.html>>__ Smart Card token on Ubuntu 14. - install the manufacturer's support software on your system and initialize the Smart Card token as per instructions (do not use PIV). - apt-get install p11-kit opensc - create directory /etc/pkcs11/modules - Add file called 'athena.module' with content

```
module: /lib64/libASEP11.so
managed: yes
```

- Verify it worked, it should resemble output below. do not continue if this does not show up.

```
$ p11-kit -l
athena: /lib64/libASEP11.so
  library-description: ASE Cryptoki
  library-manufacturer: Athena Smartcard Solutions
  library-version: 3.1
  token: IDProtect#0A50123456789
    manufacturer: Athena Smartcard Solutions
    model: IDProtect
    serial-number: 0A50123456789
    hardware-version: 1.0
    firmware-version: 1.0
    flags:
      rng
      login-required
      user-pin-initialized
      token-initialized
```

- Using pkcs11-tool, initialize your new keys. After this IDProtect Manager no longer can show your token certificates and keys, at least on version v6.23.04.

```
pkcs11-tool --module=/home/cmouse/softhsm/lib/softhsm/libsofthsm.so -l -p some-pin -k --key-type RSA:2048 -a zone-ksk
pkcs11-tool --module=/home/cmouse/softhsm/lib/softhsm/libsofthsm.so -l -p some-pin -k --key-type RSA:2048 -a zone-zsk
```

- Verify that keys are there.

```
$ pkcs11-tool --module=/lib64/libASEP11.so -l -p some-pin -O
Using slot 0 with a present token (0x0)
Public Key Object; RSA 2048 bits
```

(continues on next page)

(continued from previous page)

```

label:      zone-ksk
Usage:      encrypt, verify, wrap
Public Key Object; RSA 2048 bits
label:      zone-zsk
Usage:      encrypt, verify, wrap
Private Key Object; RSA
label:      zone-ksk
Usage:      decrypt, sign, unwrap
Private Key Object; RSA
label:      zone-zsk
Usage:      decrypt, sign, unwrap

```

- Assign the keys using

```

pdnsutil hsm assign zone rsasha256 ksk|zsk athena IDProtect#0A50123456789 pin_
↪zone-ksk/zsk

```

- Verify that everything worked, you should see valid data there.

```

pdnsutil show-zone zone

```

- Note that the physical token is pretty slow, so you have to use it as hidden master. It has been observed to produce about 1.5signatures/second.

9.9 Thanks to, acknowledgements

PowerDNS DNSSEC 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
- Vasiliy G Tolstov
- Brielle Bruns
- Evan Hunt (ISC)
- Ralf van der Enden
- Jan-Piet Mens
- Justin Clift
- Kees Monshouwer
- Aki Tuomi
- Ruben Kerkhof
- Christian Hofstaedtler
- Ruben d'Arco
- Morten Stevens
- Pieter Lexis

This list is far from complete yet ..

PER ZONE SETTINGS: DOMAIN METADATA

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.

For the BIND backend, this information is either stored in the *bind-dnssec-db* or the hybrid database, depending on your settings.

For the implementation in non-sql backends, please review your backend’s documentation.

Apart from raw SQL statements, setting domain metadata can be done with `pdnsutil set-meta` and retrieving metadata is done with `pdnsutil get-meta`.

10.1 ALLOW-AXFR-FROM

Per-zone AXFR ACLs can be stored in the domainmetadata table.

Each ACL specifies one subnet (v4 or v6), or the magical value ‘AUTO-NS’ that tries to allow all potential slaves in.

Example:

```
pdnsutil set-meta powerdns.org ALLOW-AXFR-FROM AUTO-NS 2001:db8::/48
```

Each ACL has its own row in the database:

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

To disallow all IP’s, except those explicitly allowed by domainmetadata records, add `allow-axfr-ips=` to `pdns.conf`.

10.2 API-RECTIFY

New in version 4.1.0.

This metadata item controls whether or not a zone is fully rectified on changes to the contents of a zone made through the *API*.

When the `API-RECTIFY` value is “1”, the zone will be rectified on changes. Any other other value means that it will not be rectified.

10.3 AXFR-SOURCE

The IP address to use as a source address for sending AXFR and IXFR requests.

10.4 ALLOW-DNSUPDATE-FROM, TSIG-ALLOW-DNSUPDATE, FORWARD-DNSUPDATE, SOA-EDIT-DNSUPDATE, NOTIFY-DNSUPDATE

See the documentation on *Dynamic DNS update*.

10.5 ALSO-NOTIFY

When notifying this domain, also notify this nameserver (can occur multiple times). The nameserver may have contain an optional port number. e.g.:

```
pdnsutil set-meta powerdns.org ALSO-NOTIFY 192.0.2.1:5300
pdnsutil set-meta powerdns.org ALLOW-AXFR-FROM 2001:db8:53::1
```

Or in SQL:

```
insert into domainmetadata (domain_id, kind, content) values (7, 'ALSO-NOTIFY', '192.
↪0.2.1:5300');
insert into domainmetadata (domain_id, kind, content) values (7, 'ALLOW-AXFR-FROM',
↪'2001:db8:53::1');
```

10.6 AXFR-MASTER-TSIG

Use this named TSIG key to retrieve this zone from its master, see *Provisioning signed notification and AXFR requests*.

10.7 GSS-ALLOW-AXFR-PRINCIPAL

Allow this GSS principal to perform AXFR retrieval. Most commonly it is host/something@REALM, DNS/something@REALM or user@REALM. (See *GSS-TSIG support*).

10.8 GSS-ACCEPTOR-PRINCIPAL

Use this principal for accepting GSS context. (See *GSS-TSIG support*).

10.9 IXFR

If set to 1, attempt IXFR when retrieving zone updates. Otherwise IXFR is not attempted.

10.10 LUA-AXFR-SCRIPT

Script to be used to edit incoming AXFRs, see [Modifying a slave zone using a script](#). This value will override the *lua-axfr-script* setting. Use 'NONE' to remove a global script.

10.11 NSEC3NARROW

Set to "1" to tell PowerDNS this zone operates in NSEC3 'narrow' mode. See `set-nsec3` for *pdnsutil*.

10.12 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 *pdnsutil*. Example content: "1 0 1 ab".

10.13 PRESIGNED

This zone carries DNSSEC RRSIGs (signatures), and is presigned. PowerDNS sets this flag automatically upon incoming zone transfers (AXFR) if it detects DNSSEC records in the zone. However, if you import a presigned zone using `zone2sql` or `pdnsutil load-zone` you must explicitly set the zone to be PRESIGNED. Note that PowerDNS will not be able to correctly serve the zone if the imported data is bogus or incomplete. Also see `set-presigned` in *pdnsutil*.

If a zone is presigned, the content of the metadata must be "1" (without the quotes). Any other value will not signal presignedness.

10.14 PUBLISH-CDNSKEY, PUBLISH-CDS

Whether to publish CDNSKEY and/or CDS recording defined in [RFC 7344](#).

To publish CDNSKEY records of the KSKs for the zone, set `PUBLISH-CDNSKEY` to 1.

To publish CDS records for the KSKs in the zone, set `PUBLISH-CDS` to a comma-separated list of [signature algorithm numbers](#).

This metadata can also be set using the *pdnsutil* commands `set-publish-cdnskey` and `set-publish-cds`. For an example for an [RFC 7344](#) key rollover, see the [KSK Rollover using CDS & CDNSKEY Key Rollover](#).

10.15 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. See the [DNSSEC documentation](#) for more information.

10.16 SOA-EDIT-API

On changes to the contents of a zone made through the [API](#), the SOA record will be edited according to the SOA-EDIT-API rules. These rules are the same as the [SOA-EDIT-DNSUPDATE](#) rules. If not set during zone creation, a SOA-EDIT-API metadata record is created and set to DEFAULT. If this record is removed from the backend, the default behaviour is to not do any SOA editing based on this setting. This is different from setting DEFAULT.

10.17 TSIG-ALLOW-AXFR

Allow these named TSIG keys to AXFR this zone, see *Provisioning signed notification and AXFR requests*.

10.18 TSIG-ALLOW-DNSUPDATE

This setting allows you to set the TSIG key required to do an *Dynamic DNS Update (RFC2136)*. If *GSS-TSIG* is enabled, you can put kerberos principals here as well.

10.19 Extra metadata

Through the API and on the `pdnsutil set-meta` commandline, metadata unused by PowerDNS can be added. It is mandatory to prefix this extra metadata with “X-” and the name of the external application; the API will only allow this metadata if it starts with “X-”.

DYNAMIC DNS UPDATE (RFC2136)

Starting with the PowerDNS Authoritative Server 3.4.0, DNS update support is available. There are a number of items NOT supported:

- There is no support for GSS*TSIG and SIG (TSIG is supported);
- WKS records are specifically mentioned in the RFC, we don't specifically care about WKS records;
- Anything we forgot. . . .

The implementation requires the backend to support a number of new operations. Currently, the following backends have been modified to support DNS update:

- *gmysql*
- *gpysql*
- *sqlite3*
- *goracle*
- *godbc*

11.1 Configuration options

There are two configuration parameters that can be used within the powerdns configuration file.

11.1.1 `dnsupdate`

A setting to enable/disable DNS update support completely. The default is `no`, which means that DNS updates are ignored by PowerDNS (no message is logged about this!). Change the setting to `dnsupdate=yes` to enable DNS update support. Default is `no`.

11.1.2 `allow-dnsupdate-from`

A list of IP ranges that are allowed to perform updates on any domain. The default is `127.0.0.0/8`, which means that all loopback addresses are accepted. Multiple entries can be used on this line (`allow-dnsupdate-from=198.51.100.0/8 203.0.113.2/32`). The option can be left empty to disallow everything, this then should be used in combination with the `ALLOW-DNSUPDATE-FROM` *domain metadata* setting per zone. Setting a range here and in `ALLOW-DNSUPDATE-FROM` enables updates from either address range.

11.1.3 `forward-dnsupdate`

Tell PowerDNS to forward to the master server if the zone is configured as slave. Masters are determined by the `masters` field in the `domains` table. The default behaviour is enabled (`yes`), which means that it will try to forward.

In the processing of the update packet, the `allow-dnsupdate-from` and `TSIG-ALLOW-DNSUPDATE` are processed first, so those permissions apply before the `forward-dnsupdate` is used. It will try all masters that you have configured until one is successful.

11.1.4 lua-dnsupdate-policy-script

Use this Lua script containing function `updatepolicy` to validate each update. This will **TURN OFF** all other authorization methods, and you are expected to take care of everything yourself. See [Update policy](#) for details and examples.

The semantics are that first a dynamic update has to be allowed either by the global [allow-dnsupdate-from](#) setting, or by a per-zone `ALLOW-DNSUPDATE-FROM` metadata setting.

Secondly, if a zone has a `TSIG-ALLOW-DNSUPDATE` metadata setting, that must match too.

So to only allow dynamic DNS updates to a zone based on TSIG key, and regardless of IP address, set [allow-dnsupdate-from](#) to empty, set `ALLOW-DNSUPDATE-FROM` to “0.0.0.0/0” and “:::0” and set the `TSIG-ALLOW-DNSUPDATE` to the proper key name.

Further information can be found [below](#).

11.2 Per zone settings

For permissions, a number of per zone settings are available via the [domain metadata](#).

11.2.1 ALLOW-DNSUPDATE-FROM

This setting has the same function as described in the configuration options (See [ref:above <dnsupdate-configuration-options>](#)). Only one item is allowed per row, but multiple rows can be added. An example:

```
sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata(domain_id, kind, content) values(5, 'ALLOW-
↪DNSUPDATE-FROM', '198.51.100.0/8');
sql> insert into domainmetadata(domain_id, kind, content) values(5, 'ALLOW-
↪DNSUPDATE-FROM', '203.0.113.2/32');
```

This will allow 198.51.100.0/8 and 203.0.113.2/32 to send DNS update messages for the example.org domain.

11.2.2 TSIG-ALLOW-DNSUPDATE

This setting allows you to set the TSIG key required to do an DNS update. If you have GSS-TSIG enabled, you can use Kerberos principals here. An example, using `pdnsutil` to create the key:

```
pdnsutil generate-tsig-key test hmac-md5
Create new TSIG key test hmac-md5 kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=

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

An example of how to use a TSIG key with the `nsupdate` command:

```

nsupdate <<!
server <ip> <port>
zone example.org
update add test1.example.org 3600 A 203.0.113.1
key test kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=
send
!
```

If a TSIG key is set for the domain, it is required to be used for the update. The TSIG is an alternative means of securing updates, instead of using the `ALLOW-DNSUPDATE-FROM` setting. If a TSIG key is set, and if `ALLOW-DNSUPDATE-FROM` is set, the IP(-range) of the updater still needs to be allowed via `ALLOW-DNSUPDATE-FROM`.

11.2.3 FORWARD-DNSUPDATE

See *Configuration options* <[dnsupdate-configuration-options](#)> for what it does, but per domain.

```

sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata(domain_id, kind, content) values(5, 'FORWARD-
↳DNSUPDATE', "");
```

There is no content, the existence of the entry enables the forwarding. This domain-specific setting is only useful when the configuration option *forward-dnsupdate* is set to 'no', as that will disable it globally. Using the domainmetadata setting then allows you to enable it per domain.

11.2.4 NOTIFY-DNSUPDATE

Send a notification to all slave servers after every update. This will speed up the propagation of changes and is very useful for acme verification.

```

sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata(domain_id, kind, content) values(5, 'NOTIFY-
↳DNSUPDATE', '1');
```

11.2.5 SOA-EDIT-DNSUPDATE

This configures how the soa serial should be updated. See *below*.

11.3 SOA Serial Updates

After every update, the soa serial is updated as this is required by section 3.7 of [RFC 2136](#). The behaviour is configurable via domainmetadata with the `SOA-EDIT-DNSUPDATE` option. It has a number of options listed below. If no behaviour is specified, `DEFAULT` is used.

2136, Section 3.6 defines some specific behaviour for updates of SOA records. Whenever the SOA record is updated via the update message, the logic to change the SOA is not executed.

Note: Powerdns will always use *SOA-EDIT* when serving SOA records, thus a query for the SOA record of the recently update domain, might have an unexpected result due to a `SOA-EDIT` setting.

An example:

```
sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata(domain_id, kind, content) values(5, `SOA-EDIT-
↪DNSUPDATE`, ' INCREASE');
```

This will make the SOA Serial increase by one, for every successful update.

11.3.1 SOA-EDIT-DNSUPDATE settings

These are the settings available for **SOA-EDIT-DNSUPDATE**.

- **DEFAULT**: Generate a soa serial of YYYYMMDD01. If the current serial is lower than the generated serial, use the generated serial. If the current serial is higher or equal to the generated serial, increase the current serial by 1.
- **INCREASE**: Increase the current serial by 1.
- **EPOCH**: Change the serial to the number of seconds since the EPOCH, aka unixtime.
- **SOA-EDIT**: Change the serial to whatever SOA-EDIT would provide. See [Domain metadata](#)
- **SOA-EDIT-INCREASE**: Change the serial to whatever SOA-EDIT would provide. If what SOA-EDIT provides is lower than the current serial, increase the current serial by 1.

11.4 DNS update How-to: Setup dyndns/rfc2136 with dhcpd

DNS update is often used with DHCP to automatically provide a hostname whenever a new IP-address is assigned by the DHCP server. This section describes how you can setup PowerDNS to receive DNS updates from ISC's dhcpd (version 4.1.1-P1).

11.4.1 Setting up dhcpd

We're going to use a TSIG key for security. We're going to generate a key using the following command:

```
dnssec-keygen -a hmac-md5 -b 128 -n USER dhcpdupdate
```

This generates two files (Kdhcpdupdate.*.key and Kdhcpdupdate.*.private). You're interested in the .key file:

```
# ls -l Kdhcp*
-rw----- 1 root root 53 Aug 26 19:29 Kdhcpdupdate.+157+20493.key
-rw----- 1 root root 165 Aug 26 19:29 Kdhcpdupdate.+157+20493.private

# cat Kdhcpdupdate.+157+20493.key
dhcpdupdate. IN KEY 0 3 157 FYhvwsW1ZtFZqWzsMpqbhg==
```

The important bits are the name of the key (**dhcpdupdate**) and the hash of the key (**FYhvwsW1ZtFZqWzsMpqbhg==**)

Using the details from the key you've just generated. Add the following to your dhcpd.conf:

```
key "dhcpdupdate" {
    algorithm hmac-md5;
    secret "FYhvwsW1ZtFZqWzsMpqbhg==";
};
```

You must also tell dhcpd that you want dynamic dns to work, add the following section:

```
ddns-updates on;
ddns-update-style interim;
update-static-leases on;
```

This tells **dhcpcd** to:

1. Enable Dynamic DNS
2. Which style it must use (interim)
3. Update static leases as well

For more information on this, consult the **dhcpcd.conf** manual.

Per subnet, you also have to tell **dhcpcd** which (reverse-)domain it should update and on which master domain server it is running.

```
ddns-domainname "example.org";
ddns-rev-domainname "in-addr.arpa.";

zone example.org {
    primary 127.0.0.1;
    key dhcpupdate;
}

zone 1.168.192.in-addr.arpa. {
    primary 127.0.0.1;
    key dhcpupdate;
}
```

This tells **dhcpcd** a number of things:

1. Which domain to use (**ddns-domainname “example.org”**);
2. Which reverse-domain to use (**dnssec-rev-domainname “in-addr.arpa.”**);
3. For the zones, where the primary master is located (**primary 127.0.0.1**);
4. Which TSIG key to use (**key dhcpupdate**;). We defined the key earlier.

This concludes the changes that are needed to the **dhcpcd** configuration file.

11.4.2 Setting up PowerDNS

A number of small changes are needed to powerdns to make it accept dynamic updates from **dhcpcd**.

Enabled DNS update (**RFC 2136**) support functionality in PowerDNS by adding the following to the PowerDNS configuration file (pdns.conf).

```
dnsupdate=yes
allow-dnsupdate-from=
```

This tells PowerDNS to:

1. Enable DNS update support(*dnsupdate*)
2. Allow updates from NO ip-address (“*allow-dnsupdate-from=*”)

We just told powerdns (via the configuration file) that we accept updates from nobody via the *allow-dnsupdate-from* parameter. That’s not very useful, so we’re going to give permissions per zone (including the appropriate reverse zone), via the domainmetadata table.

```
sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata(domain_id, kind, content) values(5, 'ALLOW-
↪DNSUPDATE-FROM', '127.0.0.1');
```

(continues on next page)

(continued from previous page)

```
sql> select id from domains where name='1.168.192.in-addr.arpa';
6
sql> insert into domainmetadata(domain_id, kind, content) values(6, 'ALLOW-
↪DNSUPDATE-FROM', '127.0.0.1');
```

This gives the ip '127.0.0.1' access to send update messages. Make sure you use the ip address of the machine that runs **dhcpcd**.

Another thing we want to do, is add TSIG security. This can only be done via the domainmetadata table:

```
sql> insert into tsigkeys (name, algorithm, secret) values ('dhcpcupdate', 'hmac-
↪md5', 'FYhvwsWlZtFZqWzsMpqhbg==');
sql> select id from domains where name='example.org';
5
sql> insert into domainmetadata (domain_id, kind, content) values (5, 'TSIG-ALLOW-
↪DNSUPDATE', 'dhcpcupdate');
sql> select id from domains where name='1.168.192.in-addr.arpa';
6
sql> insert into domainmetadata (domain_id, kind, content) values (6, 'TSIG-ALLOW-
↪DNSUPDATE', 'dhcpcupdate');
```

This will:

1. Add the 'dhcpcupdate' key to our PowerDNS installation
2. Associate the domains with the given TSIG key

Restart PowerDNS and you should be ready to go!

11.5 How it works

This is a short description of how DNS update messages are processed by PowerDNS.

1. The DNS update message is received. If it is TSIG signed, the TSIG is validated against the tsigkeys table. If it is not valid, Refused is returned to the requestor.
2. A check is performed on the zone to see if it is a valid zone. ServFail is returned when not valid.
3. The **dnsupdate** setting is checked. Refused is returned when the setting is 'no'.
4. If update policy Lua script is provided then next two steps are skipped.
5. If the **ALLOW-DNSUPDATE-FROM** has a value (from both domainmetadata and the configuration file), a check on the value is performed. If the requestor (sender of the update message) does not match the values in **ALLOW-DNSUPDATE-FROM**, Refused is returned.
6. If the message is TSIG signed, the TSIG keyname is compared with the TSIG keyname in domainmetadata. If they do not match, a Refused is send. The TSIG-ALLOW-DNSUPDATE domainmetadata setting is used to find which key belongs to the domain.
7. The backends are queried to find the backend for the given domain.
8. If the domain is a slave domain, the **forward-dnsupdate** option and domainmetadata settings are checked. If forwarding to a master is enabled, the message is forward to the master. If that fails, the next master is tried until all masters are tried. If all masters fail, ServFail is returned. If a master succeeds, the result from that master is returned.
9. A check is performed to make sure all updates/prerequisites are for the given zone. NotZone is returned if this is not the case.
10. The transaction with the backend is started.
11. The prerequisite checks are performed (section 3.2 of [2136](#)). If a check fails, the corresponding RCode is returned. No further processing will happen.

12. Per record in the update message, a the prescan checks are performed. If the prescan fails, the corresponding RCode is returned. If the prescan for the record is correct, the actual update/delete/modify of the record is performed. If the update fails (for whatever reason), ServFail is returned. After changes to the records have been applied, the ordername and auth flag are set to make sure DNSSEC remains working. The cache for that record is purged.
13. If there are records updated and the SOA record was not modified, the SOA serial is updated. See [SOA Serial Updates](#). The cache for this record is purged.
14. The transaction with the backend is committed. If this fails, ServFail is returned.
15. NoError is returned.

11.6 Update policy

New in version 4.1.0.

You can define a Lua script to handle DNS UPDATE message authorization. The Lua script is to contain at least function called `updatepolicy` which accepts one parameter. This parameter is an object, containing all the information for the request. To permit change, return true, otherwise return false. The script is called for each record at a time and you can approve or reject any or all.

The object has following methods available:

- `DNSName getQName()` - name to update
- `DNSName getZonename()` - zone name
- `int getQType()` - record type, it can be 255(ANY) for delete.
- `ComboAddress getLocal()` - local socket address
- `ComboAddress getRemote()` - remote socket address
- `Netmask getRealRemote()` - real remote address (or netmask if EDNS Subnet is used)
- `DNSName getTsigName()` - TSIG **key** name (you can assume it is validated here)
- `string getPeerPrincipal()` - Return peer principal name (`user@DOMAIN`, `service/machine.name@DOMAIN`, `host/MACHINE$@DOMAIN`)

There are many same things available as in recursor Lua scripts, but there is also `resolve(qname, qtype)` which returns array of records. Example:

```
resolve("www.google.com", pdns.A)
```

You can use this to perform DNS lookups. If your resolver cannot find your local records, then this will not find them either. In other words, `resolve` does not perform local lookup.

Simple example script:

```
--- This script is not suitable for production use

function strpos (haystack, needle, offset)
    local pattern = string.format("(%s)", needle)
    local i      = string.find (haystack, pattern, (offset or 0))
    return (i ~= nil and i or false)
end

function updatepolicy(input)
    princ = input:getPeerPrincipal()

    if princ == ""
    then
        return false
```

(continues on next page)

(continued from previous page)

```
end

if princ == "admin@DOMAIN" or input:getRemote():toString() == "192.168.1.1"
then
    return true
end

if (input:getQType() == pdns.A or input:getQType() == pdns.AAAA) and princ:sub(5,
↪5) == '/' and strpos(princ, "@", 0) ~= false
then
    i = strpos(princ, "@", 0)
    if princ:sub(i) ~= "@DOMAIN"
    then
        return false
    end
    hostname = princ:sub(6, i-1)
    if input:getQName():toString() == hostname .. "." or ↵
↪input:getQName():toString() == hostname .. "." .. input:getZoneName():toString()
    then
        return true
    end
end

return false
end
```

TSIG

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 *Generic SQL Backends*, they can be found in the 'tsigkeys' table. The name can be chosen freely, but the algorithm name will typically be 'hmac-md5'. Other supported algorithms are 'hmac-sha1', 'hmac-shaX' where X is 224, 256, 384 or 512. 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`)!

12.1 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. For example:

```
insert into tsigkeys (name, algorithm, secret) values ('test', 'hmac-md5', 'kp4/
↳24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/3oRSP7ys=');
select id from domains where name='powerdnssec.org';
5
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/3oRSP7ys='
```

Another of importing and activating TSIG keys into the database is using *pdnsutil*:

```
pdnsutil import-tsig-key test hmac-md5 'kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/
↳3oRSP7ys='
pdnsutil activate-tsig-key powerdnssec.org test master
```

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";
}
```

(continues on next page)

(continued from previous page)

```
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.

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

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

This can also be done using *pdnsutil*:

```
pdnsutil import-tsig-key test hmac-md5 'kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/  
↪3oRSP7ys='  
pdnsutil activate-tsig-key powerdnssec.org test slave
```

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.

12.3 GSS-TSIG support

GSS-TSIG allows authentication and authorization of DNS updates or AXFR using Kerberos with TSIG signatures.

Note: This feature is experimental and subject to change in future releases.

12.3.1 Prerequisites

- Working Kerberos environment. Please refer to your Kerberos vendor documentation on how to setup it.
- Principal (such as `DNS/<your.dns.server.name>@REALM`) in either per-user keytab or system keytab.

In particular, if something does not work, read logs and ensure that your kerberos environment is ok before filing an issue. Most common problems are time synchronization or changes done to the principal.

12.3.2 Setting up

To allow AXFR / DNS update to work, you need to configure `GSS-ACCEPTOR-PRINCIPAL` in *Per zone settings: Domain Metadata*. This will define the principal that is used to accept any GSS context requests. This *must* match to your keytab. Next you need to define one or more `GSS-ALLOW-AXFR-PRINCIPAL` entries for AXFR, or `TSIG-ALLOW-DNSUPDATE` entries for DNS update. These must be set to the exact initiator principal names you intend to use. No wildcards accepted.

GUIDES AND HOW TOS

13.1 Basic setup: configuring database connectivity

This shows you how to configure the Generic MySQL backend. This backend is called 'gmysql', and needs to be configured in `pdns.conf`. Add the following lines, adjusted for your local setup (specifically, you may not want to use the 'root' user):

```
launch=gmysql
gmysql-host=127.0.0.1
gmysql-user=root
gmysql-dbname=pdns
gmysql-password=mysecretpassword
```

Remove any earlier *launch* statements and other configuration statements for backends.

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!

Now start PowerDNS in the foreground:

```
# /usr/sbin/pdns_server --daemon=no --guardian=no --loglevel=9
(...)
Dec 30 13:40:09 About to create 3 backend threads for UDP
Dec 30 13:40:09 gmysql Connection failed: Unable to connect to database: Access_
↳denied for user 'hubert'@'localhost' to database 'pdns-non-existant'
Dec 30 13:40:09 Caught an exception instantiating a backend: Unable to launch_
↳gmysql connection: Unable to connect to database: Access denied for user 'hubert
↳'@'localhost' to database 'pdns-non-existant'
Dec 30 13:40:09 Cleaning up
Dec 30 13:40:10 Done launching threads, ready to distribute questions
```

This is as to be expected - we did not yet add anything to MySQL for PowerDNS to read from. At this point you may also see other errors which indicate that PowerDNS 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!

13.1.1 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,
```

(continues on next page)

(continued from previous page)

```

    last_check          INT DEFAULT NULL,
    type                VARCHAR(6) NOT NULL,
    notified_serial      INT UNSIGNED DEFAULT NULL,
    account              VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
    PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
    id                  BIGINT 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,
    disabled             TINYINT(1) DEFAULT 0,
    ordername            VARCHAR(255) BINARY DEFAULT NULL,
    auth                TINYINT(1) DEFAULT 1,
    PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

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

CREATE TABLE supermasters (
    ip                  VARCHAR(64) NOT NULL,
    nameserver           VARCHAR(255) NOT NULL,
    account              VARCHAR(40) CHARACTER SET 'utf8' NOT NULL,
    PRIMARY KEY (ip, nameserver)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE TABLE comments (
    id                  INT AUTO_INCREMENT,
    domain_id           INT NOT NULL,
    name                VARCHAR(255) NOT NULL,
    type                VARCHAR(10) NOT NULL,
    modified_at          INT NOT NULL,
    account              VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
    comment              TEXT CHARACTER SET 'utf8' NOT NULL,
    PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE INDEX comments_name_type_idx ON comments (name, type);
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);

CREATE TABLE domainmetadata (
    id                  INT AUTO_INCREMENT,
    domain_id           INT NOT NULL,
    kind                VARCHAR(32),
    content              TEXT,
    PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

```

(continues on next page)

(continued from previous page)

```
CREATE INDEX domainmetadata_idx ON domainmetadata (domain_id, kind);

CREATE TABLE cryptokeys (
  id                INT AUTO_INCREMENT,
  domain_id         INT NOT NULL,
  flags            INT NOT NULL,
  active            BOOL,
  content           TEXT,
  PRIMARY KEY(id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE INDEX domainidindex ON cryptokeys(domain_id);

CREATE TABLE tsigkeys (
  id                INT AUTO_INCREMENT,
  name              VARCHAR(255),
  algorithm         VARCHAR(50),
  secret            VARCHAR(255),
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

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

```
# /usr/sbin/pdns_server --daemon=no --guardian=no --loglevel=9
(...)
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
```

In a different shell, a sample query sent to the server should now return quickly without data:

```
$ dig +short www.example.com @127.0.0.1
$
```

Warning: When debugging DNS problems, don't use `host`. Please use `dig` or `drill`.

And indeed, the output in the first terminal now shows:

```
Mar 01 16:04:42 Remote 127.0.0.1 wants 'www.example.com|A', do = 0, bufsize = 1
↪1680: packetcache MISS
```

Now we need to add some records to our database (in a separate shell):

```
# mysql pdnstest
mysql> INSERT INTO domains (name, type) values ('example.com', 'NATIVE');
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'example.com','localhost admin.example.com 1 10380 3600 604800 3600','SOA'
↪',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'example.com','dns-us1.powerdns.net','NS',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'example.com','dns-eu1.powerdns.net','NS',86400,NULL);
```

(continues on next page)

(continued from previous page)

```
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'www.example.com','192.0.2.10','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'mail.example.com','192.0.2.12','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'localhost.example.com','127.0.0.1','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'example.com','mail.example.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.example.com` should be present:

```
$ dig +short www.example.com @127.0.0.1
192.0.2.10

$ dig +short example.com MX @127.0.0.1
25 mail.example.com
```

To confirm what happened, check the statistics:

```
$ /usr/sbin/pdns_control 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 hit CTRL+C in the shell where PowerDNS runs, start PowerDNS as a regular daemon, and check launch status:

On SysV systems:

```
# /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,
```

On systemd systems:

```
# systemctl start pdns.service
# systemctl status pdns.service
* pdns.service - PowerDNS Authoritative Server
   Loaded: loaded (/lib/systemd/system/pdns.service; enabled)
   Active: active (running) since Tue 2017-01-17 15:59:28 UTC; 1 months 12 days ago
     Docs: man:pdns_server(1)
           man:pdns_control(1)
           https://doc.powerdns.com
  Main PID: 24636 (pdns_server)
    CGroup: /system.slice/pdns.service
            └─24636 /usr/sbin/pdns_server --guardian=no --daemon=no --disable-
→syslog --write-pid=no

(...)
```

(continues on next page)

(continued from previous page)

```
# /usr/sbin/pdns_control 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,
```

You now have a working database driven nameserver! To convert other zones already present, see the [migration guide](#).

13.1.2 Common problems

Most problems involve PowerDNS 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 PowerDNS. Consult the MySQL documentation.

13.1.3 Typical Errors after Installing

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 PowerDNS 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 PowerDNS 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

PowerDNS did not find the `launch=bind` instruction in `pdns.conf`.

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

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

13.2 Migrating from using recursion on the Authoritative Server to using a Recursor

Recursion was removed from the Authoritative Server in version 4.1.0. This chapter discusses two scenarios and how to migrate to a new set up.

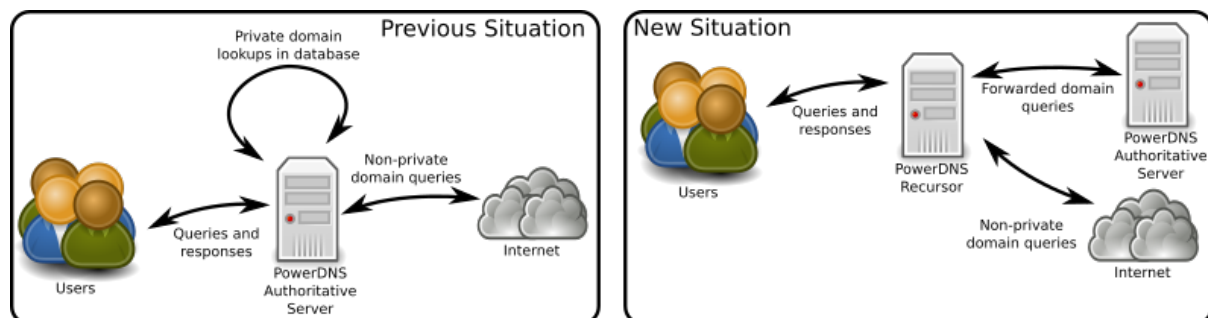
The first scenario is the one where the Authoritative Server is used as a recursor with some private domains for trusted clients. The second scenario is the one where the Authoritative Server serves publicly available domains and is a recursor for a subset of clients.

13.2.1 Scenario 1: Authoritative Server as Recursor with private zones

In this scenario, the Authoritative Server is used as a Recursor for a set of users and systems. Its database contains several private domains that are not served on the internet.

This means that migrating means that a Recursor should listen on the address the Authoritative Server. The Authoritative Server will need to listen on the local loopback interface and the Recursor should forward queries to the private domains to the Authoritative Server.

Note: These steps to require restarts and changes where services are bound to, it will inevitably lead to some down time. This guide attempts to prevent downtime to a minimum.



Migration plan

1. Remove all recursion related settings from `pdns.conf`

All settings related to recursion need to be commented out or removed from `pdns.conf` and any files included from there. These settings should be removed:

- `allow-recursion`
- `recursive-cache-ttl`
- `recursor`

2. Change the listen address and port for the Authoritative Server

To make the authoritative server listen on the local loopback address and port 5300 change the following in `pdns.conf`:

```
local-ipv6=
local-address=127.0.0.1
local-port=5300
```

3. Install and configure the PowerDNS Recursor

This is most likely an `apt-get` or `yum install` away, see the [Recursor documentation](#) for more information.

It might be possible that the Recursor can not start as the listen address is in use by the Authoritative Server, this is fine for now.

Now configure the listen addresses and ACL for the Recursor to be the same as the Authoritative Server had. The following settings should be migrated:

Authoritative Setting	Recursor Setting
local-address	local-address
local-ipv6	local-address
allow-recursion	allow-from
local-port	local-port

Now configure the recursor to forward the private domains to the Authoritative Server. This is done using the `forward-zones` setting in `recursor.conf`. The domains should be forwarded to 127.0.0.1:5300 (the new address and port of the Authoritative Server):

```
forward-zones=private.example.com=127.0.0.1:5300
forward-zones+=another.example.com=127.0.0.1:5300
# etc..
```

4. Restart the Authoritative Server and the Recursor

Restart the Authoritative Server first so its bind addresses become free for the recursor.

13.2.2 Scenario 2: Authoritative Server as Recursor for clients and serving public domains

The best way to “migrate” in this scenario is to separate the recursive service fully from the Authoritative Server. See [Dan Bernstein’s article](#) on this topic.

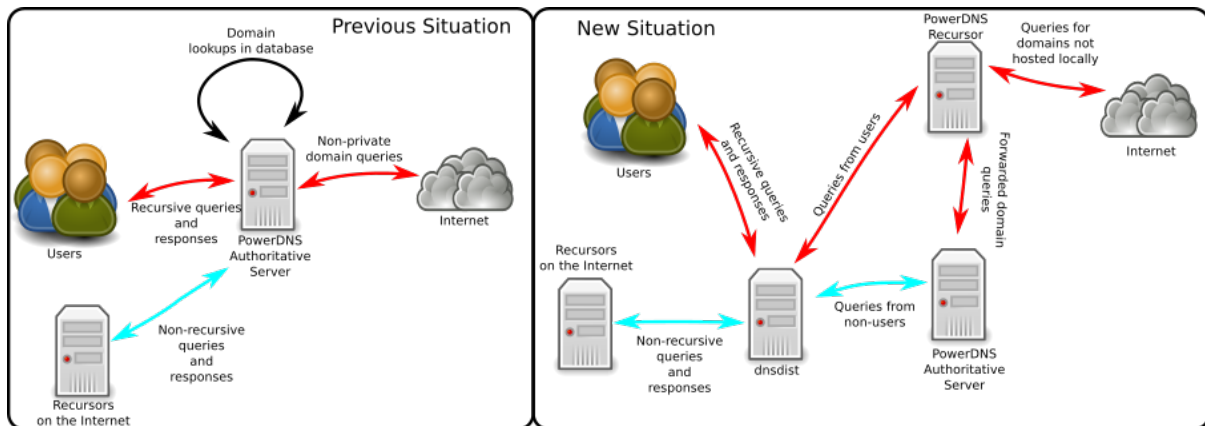
If this is not possible, this migration guide will maintain the functionality of the existing installation while allowing to upgrade.

Migration plan

1. Remove all recursion related settings from `pdns.conf`

All settings related to recursion need to be commented out or removed from `pdns.conf` and any files included from there. These settings should be removed:

- `allow-recursion`
- `recursive-cache-ttl`



- recursor

2. Change the listen address and port for the Authoritative Server

To make the authoritative server listen on the local loopback address and port 5300 change the following in `pdns.conf`:

```
local-ipv6=
local-address=127.0.0.1
local-port=5300
```

3. Install and configure the PowerDNS Recursor

This is most likely an `apt-get` or `yum install` away, see the [Recursor's Install Guide](#) for more information.

It might be possible that the Recursor can not start as the listen address is in use by the Authoritative Server, this is fine for now.

Configure the recursor to listen on the local loopback interface on a different port than the Authoritative Server. Set the following in `recursor.conf`:

```
local-address=127.0.0.1
local-port=5301
```

Now configure the recursor to forward the private domains to the Authoritative Server. This is done using the `forward-zones` setting in `recursor.conf`. The domains should be forwarded to 127.0.0.1:5300 (the new address and port of the Authoritative Server):

```
forward-zones=private.example.com=127.0.0.1:5300
forward-zones+=another.example.com=127.0.0.1:5300
# etc..
```

4. Install and configure dnsdist

`dnsdist` is a DNS loadbalancer from the people behind PowerDNS that balances DNS packets based on rules. See the [dnsdist download instructions](#) on how to install `dnsdist`.

This guide assumes `dnsdist 1.2` or `dnsdist master`.

After installing, configure `dnsdist` in `/etc/dnsdist/dnsdist.conf`. This is where several settings from the existing Authoritative Server (like listen address and recursive ACL) will be moved to.

Authoritative Setting	dnsmdist Setting
local-address	setLocal() and addLocal()
local-ipv6	setLocal() and addLocal()
local-port	setLocal() and addLocal()
allow-recursion	used in the NetmaskGroupRule()

```

setLocal('IPADDRESS:PORT')
addLocal('ANOTHERIPADDRESS:PORT')
setACL({'0.0.0.0/0', '::/0'}) -- Allow all IPs access

newServer({address='127.0.0.1:5300', pool='auth'})
newServer({address='127.0.0.1:5301', pool='recursor'})

recursive_ips = newNMG()
recursive_ips.addMask('NETWORKMASK1') -- These network masks are the ones from
↪ allow-recursion in the Authoritative Server
recursive_ips.addMask('NETWORKMASK2')

addAction(NetmaskGroupRule(recursive_ips), PoolAction('recursor'))
addAction(AllRule(), PoolAction('auth'))

```

This configuration will route all queries from the netmasks that are allowed to do recursion to the Recursor and all other queries to the Authoritative Server.

5. Restart the Authoritative Server, the Recursor and dnsmdist

Restart the Authoritative Server first so its bind addresses become free for the recursor.

13.3 Running Virtual Instances

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

To generate additional PowerDNS instances, create a `pdns-NAME.conf` in your configuration directory (usually `/etc/powerdns`), where `NAME` is the name of your virtual configuration.

Following one of the following instructions, PowerDNS will read its configuration from the `pdns-NAME.conf` instead of `pdns.conf`.

13.3.1 Starting virtual instances with Sysv init-scripts

Symlink the `init.d` script `pdns` to `pdns-NAME`, where `NAME` is the name of your virtual configuration.

Warning: `NAME` must not contain a `'-'` as this will confuse the script.

Internally, the `init` script calls the binary with the `config-name` option set to `name`, setting in motion the loading of separate configuration files.

When you launch a virtual instance of PowerDNS, the `pid-file` is saved inside `socket-dir` as `pdns-name.pid`.

Warning: Be aware however that the `init.d force-stop` will kill all PowerDNS instances!

13.3.2 Starting virtual instances with systemd

With systemd it is as simple as calling the correct service instance. Assuming your instance is called `myinstance` and `pdns-myinstance.conf` exists in the configuration directory, the following command will start the service:

```
systemctl start pdns@myinstance.service
```

Similarly you can enable it at boot:

```
systemctl enable pdns@myinstance.service
```

13.4 Using ALIAS records

The ALIAS record provides a way to have CNAME-like behaviour on the zone apex.

In order to correctly serve ALIAS records in PowerDNS Authoritative Server 4.1.0 or higher, set the *resolver* setting to an existing resolver and enable *expand-alias*:

```
resolver=[::1]:5300
expand-alias=yes
```

Note: If *resolver* is unset, ALIAS expansion is disabled!

Then add the ALIAS record to your zone apex. e.g.:

```
$ORIGIN example.net
$TTL 1800

@ IN SOA ns1.example.net. hostmaster.example.net. 2015121101 1H 15 1W 2H

@ IN NS ns1.example.net.

@ IN ALIAS mywebapp.paas-provider.net.
```

When the authoritative server receives a query for the A-record for `example.net`, it will resolve the A record for `mywebapp.paas-provider.net` and serve an answer for `example.net` with that A record.

When a zone containing ALIAS records is transferred over AXFR, the *outgoing-axfr-expand-alias* setting controls the behaviour of ALIAS records. When set to 'no' (the default), ALIAS records are sent as-is (RRType 65401 and a DNSName in the RDATA) in the AXFR. When set to 'yes', PowerDNS will lookup the A and AAAA records of the name in the ALIAS-record and send the results in the AXFR.

Set *outgoing-axfr-expand-alias* to 'yes' if your slaves don't understand ALIAS or should not look up the addresses themselves. Note that slaves will not automatically follow changes in those A/AAAA records unless you AXFR regularly.

Note: The *expand-alias* setting does not exist in PowerDNS Authoritative Server 4.0.x. Hence, ALIAS records are always expanded on a direct A or AAAA query.

13.4.1 ALIAS and DNSSEC

Starting with the PowerDNS Authoritative Server 4.0.0, DNSSEC 'washing' of ALIAS records is supported on AXFR (**not** on live-signing). Set *outgoing-axfr-expand-alias* to 'yes' and enable DNSSEC for the zone on the master. PowerDNS will sign the A/AAAA records during the AXFR.

13.5 KSK Rollover

Before attempting a KSK rollover, please read [RFC 6581 “DNSSEC Operational Practices, Version 2”](#), [section 4](#) carefully to understand the terminology, actions and timelines (TTL and RRSIG expiry) involved in rolling a KSK.

This How To describes the “Double-Signature Key Signing Key Rollover” from the above mentioned RFC.

To start the rollover, add an **active** new KSK to the zone (example.net in this case):

```
pdnsutil add-zone-key example.net ksk active
```

Note that a key with same algorithm as the KSK to be replaced should be created, as this is not an algorithm roll over.

If this zone is of the type ‘MASTER’, increase the SOA serial. The rollover is now in the “New KSK” stage. Retrieve the DS record(s) for the new KSK:

```
pdnsutil show-zone example.net
```

And communicate this securely to your registrar/parent zone. Now wait until the new DS is published in the parent zone and at least the TTL for the DS records has passed. The rollover is now in the “DS Change” state and can continue to the “DNSKEY Removal” stage by actually deleting the old KSK.

Note: The key-id for the old KSK is shown in the output of `pdnsutil show-zone example.net`.

```
pdnsutil remove-zone-key example.net KEY-ID
```

The rollover is now complete.

13.6 KSK Rollover using CDS & CDNSKEY Key Rollover

If the upstream registry supports [RFC 7344](#) key rollovers you can use several *pdnsutil* commands to do this rollover. This HowTo follows the rollover example from the RFCs [Appendix B](#).

We assume the zone name is example.com and is already DNSSEC signed.

Start by adding a new KSK to the zone: `pdnsutil add-zone-key example.com ksk 2048 inactive`. The “inactive” means that the key is not used to sign any ZSK records. This limits the size of ANY and DNSKEY responses.

Publish the CDS records: `pdnsutil set-publish-cds example.com`, these records will tell the parent zone to update its DS records. Now wait for the DS records to be updated in the parent zone.

Once the DS records are updated, do the actual key-rollover: `pdnsutil activate-zone-key example.com new-key-id` and `pdnsutil deactivate-zone-key example.com old-key-id`. You can get the new-key-id and old-key-id by listing them through `pdnsutil show-zone example.com`.

After the rollover, wait *at least* until the TTL on the DNSKEY records have expired so validating resolvers won’t mark the zone as BOGUS. When the wait is over, delete the old key from the zone: `pdnsutil remove-zone-key example.com old-key-id`. This updates the CDS records to reflect only the new key.

Wait for the parent to pick up on the CDS change. Once the upstream DS records show only the DS records for the new KSK, you may disable sending out the CDS responses: `pdnsutil unset-publish-cds example.com`.

Done!

13.7 ZSK Rollover

This how to describes the way to roll a ZSK that is not a secure entryptpoint (a ZSK that is not tied to a DS record in the parent zone) using the “[RFC 6781 Pre-Publish Zone Signing Key Rollover](#)” method. The documentation linked above also lists the minimum time between stages. **PLEASE READ THAT DOCUMENT CAREFULLY**

First, create a new inactive ZSK for the zone (if one already exists, you can skip this step), we add an ECDSA 256 bit key (algorithm 13) here:

```
pdnsutil add-zone-key example.net zsk inactive ecdsa256
```

You are now almost at the “new DNSKEY”-stage of the rollover, if the zone is of type ‘MASTER’ you’ll need to update the SOA serial in the database and wait for the slaves to pickup the zone change.

To change the RRSIGs on your records, the new key must be made active. Note: you can get the key-ids with `pdnsutil show-zone example.net`:

```
pdnsutil activate-zone-key example.net new-key-id
pdnsutil deactivate-zone-key example.net previous-key-id
```

Again, if this is a ‘MASTER’-zone, update the SOA serial. You are now at the “new RRSIGs” stage of the roll over.

The last step is to remove the old key from the completely:

```
pdnsutil remove-zone-key example.net previous-key-id
```

Don’t forget to update the SOA serial for ‘MASTER’ zones. The rollover is now at the “DNSKEY removal” stage and complete.

13.8 Adding new DNS record types

Here are the full descriptions on how we added the TLSA record type to all PowerDNS products, with links to the actual source code.

First, define the TLSARecordContent class in `dnsrecords.hh`:

```
class TLSARecordContent : public DNSRecordContent
{
public:
    includeboilerplate(TLSA)

private:
    uint8_t d_certusage, d_selector, d_matchtype;
    string d_cert;
};
```

The `includeboilerplate(TLSA)` macro generates the four methods that do everything PowerDNS would ever want to do with a record:

- read TLSA records from zonefile format
- write out a TLSA record in zonefile format
- read a TLSA record from a packet
- write a TLSA record to a packet

The actual parsing code:

```
boilerplate_conv(TLSA, 52,
    conv.xfr8BitInt(d_certusage);
    conv.xfr8BitInt(d_selector);
    conv.xfr8BitInt(d_matchtype);
    conv.xfrHexBlob(d_cert, true);
)
```

This code defines the TLSA rrtype number as 52. Secondly, it says there are 3 eight bit fields for Certificate Usage, Selector and Match type. Next, it defines that the rest of the record is the actual certificate (hash). ‘conv’ methods are supplied for all DNS data types in use.

Now add `TLSARecordContent::report()` to ``reportOtherTypes()` <<https://github.com/PowerDNS/pdns/blob/5a3409cbb4314b84f1171a69c7337386568fa886/pdns/dnsrecords.cc#L594>>‘__.

And that’s it. For completeness, add TLSA and 52 to the QType enum in ``qtype.hh` <<https://github.com/PowerDNS/pdns/blob/5a3409cbb4314b84f1171a69c7337386568fa886/pdns/qtype.hh#L116>>‘__, which makes it easier to refer to the TLSA record in code if so required.

BACKENDS

The following table describes the supported backends and some of their capabilities.

Name	Native	Master	Slave	Super slave	Auto serial	<i>DNSSEC</i>	Launch
<i>BIND</i>	Yes	Yes	Yes	Experimental	No	Yes	bind
<i>Generic Mysql</i>	Yes	Yes	Yes	Yes	Yes	Yes	gmysql
<i>Generic ODBC</i>	Yes	Yes	Yes	Yes	Yes	Yes	godbc
<i>Generic Oracle</i>	Yes	Yes	Yes	Yes	Yes	Yes	goracle
<i>Generic Postgresql</i>	Yes	Yes	Yes	Yes	Yes	Yes	gpgsql
<i>Generic SQLite3</i>	Yes	Yes	Yes	Yes	Yes	Yes	gsqlite3
<i>GeoIP</i>	Yes	No	No	No	No	Yes	geoip
<i>LDAP</i>	Yes	No	No	No	No	No	ldap
<i>MyDNS</i>	Yes	No	No	No	No	No	mydns
<i>OpenDBX</i>	Yes	Yes	Yes	Yes	No	No	opendbx
<i>Oracle</i>	Yes	Yes	Yes	Yes	Yes	Yes	oracle
<i>Pipe</i>	Yes	No	No	No	No	Partial	pipe
<i>Random</i>	Yes	No	No	No	No	Partial	random
<i>Remote</i>	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	remote
<i>TinyDNS</i>	Yes	Yes	No	No	No	Partial	tinydns

All the generic SQL backends have similar functionality, apart from the database they communicate with. These backends have *features unique* to the generic SQL backends.

14.1 Bind zone file backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Experimental
- Autoserial: No
- DNSSEC: Yes
- Disabled data: No
- Comments: No
- Module name: bind
- Launch: bind

The BindBackend started life as a demonstration of the versatility of PowerDNS 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 PowerDNS native configuration.

14.1.1 Configuration Parameters

`bind-config`

Location of the Bind configuration file to parse.

PowerDNS does not support every directive supported by Bind. It supports the following blocks and directives:

- **options**
 - `directory`
 - `also-notify`
- **zone**
 - `file`
 - `type`
 - `masters`
 - `also-notify`

`bind-check-interval`

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

`bind-dnssec-db`

Filename to store and access our DNSSEC metadatabase, empty for none. To slave DNSSEC-enabled domains (where the RRSIGS are in the AXFR), a `bind-dnssec-db` is required. This is because the *PRESIGNED* domain metadata is set during the zonetransfer.

`bind-hybrid`

Store DNSSEC keys and metadata storage in an other backend. See the *Hybrid BIND-mode operation* documentation.

`bind-ignore-broken-records`

Setting this option to `yes` makes PowerDNS ignore out of zone records when loading zone files.

14.1.2 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.

14.1.3 pdns_control commands

bind-add-zone <domain> <filename>

Add zone `domain` from `filename` to PowerDNS's bind backend. Zone will be loaded at first request.

Note: This does not add the zone to the *bind-config* file.

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.

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.

reload

All zones with a changed timestamp are reloaded at the next incoming query for them.

14.1.4 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`.

14.1.5 Master/slave/native configuration

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.

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.

Native

PowerDNS has the concept of “native” zones that have the `type native;` in the BIND configuration file. These zones are neither a master (no notifies are sent) nor a slave zone (it will never be AXFR'd in). This means that the replication mechanism for these zone is not AXFR but out of band, e.g. using `rsync`. Changes to native zones are picked up in the same way as master and slave zones, see [Operation](#).

Native zones in the BIND backend are supported since version 4.1.0 of the PowerDNS Authoritative Server.

note: Any zone with no `type` set (an error in BIND) is assumed to be native.

14.2 Generic SQL Backends

The generic SQL backends (like `gmysql`, `gpgsql` and `godbc`) are backends with easily configurable SQL statements, allowing you to graft PowerDNS on any SQL 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.

Note: A root zone or record should have a name of '.' (no quotes). This is the only exception to the 'no terminating dot in SQL storage' rule.

14.2.1 Basic functionality

All domains in the generic SQL backends have a 'type' field that describes the [DNS Modes of Operation](#).

Native operation

To add a domain, issue the following:

```
INSERT INTO domains (name, type) VALUES ('example.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.

Slave operation

These backends are fully slave capable. To become a slave of the 'example.com' domain, execute this:

```
INSERT INTO domains (name, master, type) VALUES ('example.com', '198.51.100.6',  
↪ 'SLAVE');
```

And wait a while for PowerDNS to pick up the addition - which happens within one minute (this is determined by the *slave-cycle-interval* setting). There is no need to inform PowerDNS 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 example.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 'example.com'
Apr 09 13:35:30 AXFR done for 'example.com'
Apr 09 13:35:30 [gPgSQLBackend] Closing connection
```

From now on, PowerDNS is authoritative for the ‘example.com’ zone and will respond accordingly for queries within that zone.

Periodically, PowerDNS 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.

PowerDNS has support for multiple masters per zone, separate master IP addresses by commas:

```
INSERT INTO domains (name, master, type) VALUES ('example.com', '198.51.100.6, ↵
↵2001:0DB8:15:4AF::4', 'SLAVE');
```

Superslave operation

To configure a supermaster with IP address 203.0.113.53 which lists this installation as ‘autoslave.example.com’, issue the following:

```
INSERT INTO supermasters VALUES ('203.0.113.53', 'autoslave.example.com', 'internal
↵');
```

From now on, valid notifies from 203.0.113.53 that list a NS record containing ‘autoslave.example.com’ will lead to the provisioning of a slave domain under the account ‘internal’. See *Supermaster: automatic provisioning of slaves* for details.

Master operation

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

```
INSERT INTO domains (name, type) VALUES ('example.com', 'MASTER');
```

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

Disabled data

PowerDNS understands the notion of disabled records. They are marked by setting “disabled” to 1 (for PostgreSQL: `true`). By extension, when the SOA record for a domain is disabled, the entire domain is considered to be disabled.

Effects: the record (or domain, respectively) will not be visible to DNS clients. The REST API will still see the record (or domain). Even if a domain is disabled, slaving still works. Slaving considers a disabled domain to have a serial of 0; this implies that a slaved domain will not stay disabled.

Autoserial

The autoserial functionality makes PowerDNS generate the SOA serial when the SOA serial is set to 0 in the database. The serial in SOA responses is set to what's provided by `zone-lastchange-query`. By default, this is the highest value of the `change_date` field in the "records" table).

14.2.2 Handling DNSSEC signed zones

To enable DNSSEC processing, the `backend-dnssec` option must be set to 'yes'.

Rules for filling out DNSSEC fields

Two additional fields in the 'records' table are important: 'auth' and 'ordname'. These fields are set correctly on an incoming zone transfer, and also by running `pdnsutil rectify-zone`.

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/NSEC3 mode, the ordname field should be NULL for any glue but filled in for all delegation NS records and all authoritative records. In NSEC3 opt-out mode, ordname is NULL for any glue and insecure delegation NS records, but filled in for secure delegation NS records and all authoritative records.

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. `pdnsutil hash-zone-record zone record` can be used to calculate this hash.

In addition, PowerDNS fully supports empty non-terminals. If you have a zone `example.com`, and a host `a.b.c.example.com` in it, `rectify-zone` (and the AXFR client code) will insert `b.c.example.com` and `c.example.com` in the records table with type NULL (SQL NULL, not 'NULL'). Having these entries provides several benefits. We no longer reply NXDOMAIN for these shorter names (this was an RFC violation but not one that caused trouble). But more importantly, to do NSEC3 correctly, we need to be able to prove existence of these shorter names. The type=NULL records entry gives us a place to store the NSEC3 hash of these names.

If your frontend does not add empty non-terminal names to records, you will get DNSSEC replies of 3.1-quality, which has served many people well, but might lead to issues in the future.

14.2.3 Queries

From version 4.0.0 onward, the generic SQL backends use prepared statements for their queries. Before 4.0.0, queries were expanded using the C function 'snprintf' which implies that substitutions are performed on the basis of %-placeholders.

To see the default queries for a backend, run `pdns_server --no-config --launch=BACKEND --config`.

Regular Queries

For regular operation, several queries are used for record-lookup. These queries must return the following fields in order:

- **content:** This is the ‘right hand side’ of a DNS record. For an A record, this is the IP address for example.
- **ttl:** TTL of this record, in seconds. Must be a positive integer, no checking is performed.
- **prio:** For MX and SRV records, this should be the priority of the record specified.
- **type:** The ASCII representation of the qtype of this record. Examples are ‘A’, ‘MX’, ‘SOA’, ‘AAAA’. Make sure that this field returns an exact answer - PowerDNS won’t recognise ‘A ‘ as ‘A’. This can be achieved by using a VARCHAR instead of a CHAR.
- **domain_id:** Unique identifier for this domain. This id must be unique across all backends. Must be a positive integer.
- **disabled:** Boolean, if set to true, this record is hidden from DNS clients, but can still be modified from the REST API. See [Disabled data](#).
- **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!
- **auth:** A boolean describing if PowerDNS is authoritative for this record (DNSSEC)

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

- **basic-query:** This is the most used query, needed for doing 1:1 lookups of qtype/name values.
- **id-query:** Used for doing lookups within a domain.
- **any-query:** For doing ANY queries. Also used internally.
- **any-id-query:** For doing ANY queries within a domain. Also used internally.
- **list-query:** For doing AXFRs, lists all records in the zone. Also used internally.
- **list-subzone-query:** For doing RFC 2136 DNS Updates, lists all records below a zone.
- **search-records-query:** To search for records on name and content.

DNSSEC queries

These queries are used by e.g. `pdnsutil rectify-zone`. Make sure to read [Rules for filling out DNSSEC fields](#) if you wish to calculate ordername and auth without using `pdns-rectify`.

- **insert-empty-non-terminal-order--query:** Insert empty non-terminal in zone.
- **delete-empty-non-terminal-query:** Delete an empty non-terminal in a zone.
- **remove-empty-non-terminals-from-zone-query:** remove all empty non-terminals from zone.
- **get-order-first-query:** DNSSEC Ordering Query, first.
- **get-order-before-query:** DNSSEC Ordering Query, before.
- **get-order-after-query:** DNSSEC Ordering Query, after.
- **get-order-last-query:** DNSSEC Ordering Query, last.
- **update-ordername-and-auth-query:** DNSSEC update ordername and auth for a qname query.
- **update-ordername-and-auth-type-query:** DNSSEC update ordername and auth for a rrset query.
- **nullify-ordername-and-update-auth-query:** DNSSEC nullify ordername and update auth for a qname query.
- **nullify-ordername-and-update-auth-type-query:** DNSSEC nullify ordername and update auth for a rrset query.

Domain and zone manipulation

- `is-our-domain-query`: Checks if the domain (either id or name) is in the ‘domains’ table. This query is run before any other (possibly heavy) query.
- `insert-zone-query`: Add a new domain. This query also requires the type, masters and account fields
- `update-kind-query`: Called to update the type of domain.
- `delete-zone-query`: Called to delete all records of a zone. Used before an incoming AXFR.
- `delete-domain-query`: Called to delete a domain from the domains-table.
- `get-all-domains-query`: Used to get information on all active domains.
- `info-zone-query`: Called to retrieve (nearly) all information for a domain.
- `insert-record-query`: Called during incoming AXFR.
- `update-account-query`: Set the account for a domain.
- `delete-names-query`: Called to delete all records of a certain name.
- `delete-rrset-query`: Called to delete an RRset based on domain_id, name and type.
- `get-all-domain-metadata-query`: Get all *domain metadata* for a domain.
- `get-domain-metadata-query`: Get a single piece of *domain metadata*.
- `clear-domain-metadata-query`: Delete a single entry of domain metadata.
- `clear-domain-all-metadata-query`: Remove all domain metadata for a domain.
- `set-domain-metadata-query`: Add domain metadata for a zone.
- `add-domain-key-query`: Called to add a cryptokey to a domain.
- `list-domain-keys-query`: Called to get all cryptokeys for a domain.
- `activate-domain-key-query`: Called to set a cryptokey to active.
- `deactivate-domain-key-query`: Called to set a cryptokey to inactive.
- `clear-domain-all-keys-query`: Called to remove all DNSSEC keys for a zone.
- `remove-domain-key-query`: Called to remove a crypto key.

Master/slave queries

These queries are used to manipulate the master/slave information in the database. Most installations will have zero need to change the following queries.

On masters

- `info-all-master-query`: Called to get data on all domains for which the server is master.
- `update-serial-query`: Called to update the last notified serial of a master domain.
- `zone-lastchange-query`: Called to determine the last change to a zone, used for autoserial.

On slaves

- `info-all-slaves-query`: Called to retrieve all slave domains.
- `master-zone-query`: Called to determine the master of a zone.
- `update-lastcheck-query`: Called to update the last time a slave domain was successfully checked for freshness.

- `update-master-query`: Called to update the master address of a domain.

On superslaves

- `supermaster-query`: Called to determine if a certain host is a supermaster for a certain domain name.
- `supermaster-name-to-ips`: Called to the IP and account for a supermaster.

TSIG

- `get-tsig-key-query`: Called to get the algorithm and secret from a named TSIG key.
- `get-tsig-keys-query`: Called to get all TSIG keys.
- `set-tsig-key-query`: Called to set the algorithm and secret for a named TSIG key.
- `delete-tsig-key-query`: Called to delete a named TSIG key.

Comment queries

For listing/modifying comments.

- `list-comments-query`: Called to get all comments in a zone. Returns fields: `domain_id`, `name`, `type`, `modified_at`, `account`, `comment`.
- `insert-comment-query`: Called to create a single comment for a specific RRSet. Given fields: `domain_id`, `name`, `type`, `modified_at`, `account`, `comment`
- `delete-comment-rrset-query`: Called to delete all comments for a specific RRset. Given fields: `domain_id`, `name`, `type`
- `delete-comments-query`: Called to delete all comments for a zone. Usually called before deleting the entire zone. Given fields: `domain_id`
- `search-comments-query`: Called to search for comment by name or content.

Specifying queries

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

```
gmysql-basic-query=SELECT content,ttl,prio,type,domain_id,disabled,name,auth FROM ↵
↵records WHERE disabled=0 and type=? and name=?
```

Queries can span multiple lines, like this:

```
gmysql-basic-query=SELECT content,ttl,prio,type,domain_id,disabled,name,auth \
FROM records WHERE disabled=0 and type=? and name=?
```

14.3 Generic MySQL backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserial: Yes

- Case: All lower
- DNSSEC: Yes (set `gmysql-dnssec`)
- Disabled data: Yes
- Comments: Yes
- Module name: `gmysql`
- Launch name: `gmysql`

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 schema is included at the bottom of this page. *Using zone2sql* with the `--gmysql` flag also assumes this layout is in place. For full migration notes, please see *Migrating to PowerDNS*. This schema contains all elements needed for master, slave and superslave operation.

When using the InnoDB storage engine, we suggest adding foreign key constraints to the tables in order to automate deletion of records, key material, and other information upon deletion of a domain from the domains table. The following SQL does the job:

```
/*
Using this SQL causes Mysql to create foreign keys on your database. This will
make sure that no records, comments or keys exists for domains that you already
removed. This is not enabled by default, because we're not sure what the
consequences are from a performance point of view. If you do have feedback,
please let us know how this effects your setup.

Please note that it's not possible to apply this, before you cleaned up your
database, as the foreign keys do not exist.
*/
ALTER TABLE records ADD CONSTRAINT `records_domain_id_ibfk` FOREIGN KEY (`domain_
↪id`) REFERENCES `domains` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE comments ADD CONSTRAINT `comments_domain_id_ibfk` FOREIGN KEY (`domain_
↪id`) REFERENCES `domains` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE domainmetadata ADD CONSTRAINT `domainmetadata_domain_id_ibfk` FOREIGN
↪KEY (`domain_id`) REFERENCES `domains` (`id`) ON DELETE CASCADE ON UPDATE
↪CASCADE;
ALTER TABLE cryptokeys ADD CONSTRAINT `cryptokeys_domain_id_ibfk` FOREIGN KEY
↪(`domain_id`) REFERENCES `domains` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

14.3.1 Using MySQL replication

To support NATIVE domains, the `binlog_format` for the MySQL replication **must** be set to MIXED or ROW to prevent differences in data between replicated servers. See “[Setting The Binary Log Format](#)” for more information.

14.3.2 Settings

`gmysql-host`

Host (ip address) to connect to. Mutually exclusive with *gmysql-socket*.

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

gmysql-port

The port to connect to on *gmysql-host*. Default: 3306.

gmysql-socket

Connect to the UNIX socket at this path. Mutually exclusive with *gmysql-host*.

gmysql-dbname

Name of the database to connect to. Default: “powerdns”.

gmysql-user

User to connect as. Default: “powerdns”.

gmysql-group

Group to connect as. Default: “client”.

gmysql-password

The password to for *gmysql-user*.

gmysql-dnssec

Enable DNSSEC processing for this backend. Default: no.

gmysql-innodb-read-committed

Use the InnoDB READ-COMMITTED transaction isolation level. Default: yes.

gmysql-timeout

The timeout in seconds for each attempt to read from, or write to the server. A value of 0 will disable the timeout. Default: 10

14.3.3 Default 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 UNSIGNED DEFAULT NULL,
  account     VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE UNIQUE INDEX name_index ON domains(name);
```

(continues on next page)

(continued from previous page)

```
CREATE TABLE records (  
  id BIGINT 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,  
  disabled TINYINT(1) DEFAULT 0,  
  ordername VARCHAR(255) BINARY DEFAULT NULL,  
  auth TINYINT(1) DEFAULT 1,  
  PRIMARY KEY (id)  
) Engine=InnoDB CHARACTER SET 'latin1';  
  
CREATE INDEX nametype_index ON records(name, type);  
CREATE INDEX domain_id ON records(domain_id);  
CREATE INDEX ordername ON records (ordername);  
  
CREATE TABLE supermasters (  
  ip VARCHAR(64) NOT NULL,  
  nameserver VARCHAR(255) NOT NULL,  
  account VARCHAR(40) CHARACTER SET 'utf8' NOT NULL,  
  PRIMARY KEY (ip, nameserver)  
) Engine=InnoDB CHARACTER SET 'latin1';  
  
CREATE TABLE comments (  
  id INT AUTO_INCREMENT,  
  domain_id INT NOT NULL,  
  name VARCHAR(255) NOT NULL,  
  type VARCHAR(10) NOT NULL,  
  modified_at INT NOT NULL,  
  account VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,  
  comment TEXT CHARACTER SET 'utf8' NOT NULL,  
  PRIMARY KEY (id)  
) Engine=InnoDB CHARACTER SET 'latin1';  
  
CREATE INDEX comments_name_type_idx ON comments (name, type);  
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);  
  
CREATE TABLE domainmetadata (  
  id INT AUTO_INCREMENT,  
  domain_id INT NOT NULL,  
  kind VARCHAR(32),  
  content TEXT,  
  PRIMARY KEY (id)  
) Engine=InnoDB CHARACTER SET 'latin1';  
  
CREATE INDEX domainmetadata_idx ON domainmetadata (domain_id, kind);  
  
CREATE TABLE cryptokeys (  
  id INT AUTO_INCREMENT,  
  domain_id INT NOT NULL,  
  flags INT NOT NULL,  
  active BOOL,  
  content TEXT,
```

(continues on next page)

(continued from previous page)

```
PRIMARY KEY(id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE INDEX domainidindex ON cryptokeys(domain_id);

CREATE TABLE tsigkeys (
  id          INT AUTO_INCREMENT,
  name        VARCHAR(255),
  algorithm   VARCHAR(50),
  secret      VARCHAR(255),
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

14.4 Generic ODBC Backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserail: Yes
- Case: All lower
- DNSSEC: Yes
- Disabled data: Yes
- Comments: Yes
- Module name: godbc
- Launch name: godbc

The Generic ODBC Backend (godbc) is a child of the Generic SQL (gsql) backend, similar to the gmysql and gpgsql backends. It uses [UnixODBC](#) and installed drivers to connect to the databases supported by said drivers.

Warning: When there is a more specific generic sql backend (like goracle or gmysql), it is highly recommended to use that backend instead!

14.4.1 Enabling the backend

When building PowerDNS yourself, append godbc to `--with-modules` or `--with-dynmodules`. It is expected that most pre-built packages contain this backend or be separately installable.

14.4.2 Configuration Parameters

This section only details the configuration of PowerDNS for use with ODBC. For ODBC related configuration, please see [UnixODBC website/documentation](#) and the documentation for the driver you intend to use.

godbc-datasource

- String
- Default: PowerDNS

The datasource (DSN) to use. This must be configured in the `odbc.ini` file, usually found in `/etc/`, but this depends on your local setup.

godbc-username

- String
- Default: powerdns

The user to connect to the datasource.

godbc-password

- String
- Default is empty

The password to connect with the datasource.

14.4.3 Connecting to Microsoft SQL Server

Note: In order to connect to Microsoft SQL Server, you will need at least version 3.2.0 of UnixODBC. FreeTDS has been tested with versions 0.91 and 0.95.

Install the [FreeTDS](#) driver for UnixODBC, either by compiling or getting it from our distribution's repository and configure your `/etc/odbcinst.ini` with the driver, e.g.:

```
[FreeTDS]
Description=v0.95.8 with protocol v7.1
Driver=/usr/local/lib/libtdsodbc.so
UsageCount=1
```

And add the datasource to your `/etc/odbc.ini`, e.g.:

```
[pdns1]
Driver=FreeTDS
Trace=No
Server=server.example.net
Port=1433
Database=pdns-1
TDS_Version=7.1
```

(For our tests, we add `ClientCharset=UTF-8` as well. YMMV.)

You can now test the connection with `isql pdns1 USERNAME PASSWORD`.

Loading the schema into the database

For convenience, a schema for MS SQL Server has been created: (Note: This schema can also be found in the PowerDNS source as `modules/godbcbackend/schema.mssql.sql`).

```

CREATE TABLE domains (
  id          INT IDENTITY,
  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 UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
  id          INT IDENTITY,
  domain_id   INT DEFAULT NULL,
  name        VARCHAR(255) DEFAULT NULL,
  type        VARCHAR(10) DEFAULT NULL,
  content     VARCHAR(MAX) DEFAULT NULL,
  ttl         INT DEFAULT NULL,
  prio        INT DEFAULT NULL,
  change_date INT DEFAULT NULL,
  disabled    BIT DEFAULT 0,
  ordername   VARBINARY(255) DEFAULT NULL,
  auth        BIT DEFAULT 1,
  PRIMARY KEY (id)
);

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

CREATE TABLE supermasters (
  ip          VARCHAR(64) NOT NULL,
  nameserver  VARCHAR(255) NOT NULL,
  account     VARCHAR(40) NOT NULL,
  PRIMARY KEY (ip, nameserver)
);

CREATE TABLE comments (
  id          INT IDENTITY,
  domain_id   INT NOT NULL,
  name        VARCHAR(255) NOT NULL,
  type        VARCHAR(10) NOT NULL,
  modified_at INT NOT NULL,
  account     VARCHAR(40) NOT NULL,
  comment     VARCHAR(MAX) NOT NULL,
  PRIMARY KEY (id)
);

CREATE INDEX comments_domain_id_idx ON comments (domain_id);
CREATE INDEX comments_name_type_idx ON comments (name, type);
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);

CREATE TABLE domainmetadata (
  id          INT IDENTITY,
  domain_id   INT NOT NULL,
  kind        VARCHAR(32),
  content     VARCHAR(MAX),

```

(continues on next page)

(continued from previous page)

```
PRIMARY KEY (id)
);

CREATE INDEX domainmetadata_idx ON domainmetadata (domain_id, kind);

CREATE TABLE cryptokeys (
  id          INT IDENTITY,
  domain_id   INT NOT NULL,
  flags       INT NOT NULL,
  active      BIT,
  content      VARCHAR(MAX),
  PRIMARY KEY(id)
);

CREATE INDEX domainidindex ON cryptokeys(domain_id);

CREATE TABLE tsigkeys (
  id          INT IDENTITY,
  name        VARCHAR(255),
  algorithm   VARCHAR(50),
  secret      VARCHAR(255),
  PRIMARY KEY(id)
);

CREATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

Load this into the database as follows:

```
cat schema.mssql.sql | tr '\n' ' ' | isql pdns1 USERNAME PASSWORD -b.
```

Loading records into the database

Loading records is the same as with any SQL backend, just add them using SQL-queries. Should you want to use *zone2sql*, use the `--sqlite` option for correctly formatted SQL.

Configuring PowerDNS

Add the options required to your `pdns.conf`:

```
launch=godbc
godbc-datasource=pdns1
godbc-username=USERNAME
godbc-password=PASSWORD
```

Now restart PowerDNS and you're done. Just don't forget to add zones and records to the database.

Possible issues

It might be that you need to compile FreeTDS with the `--tds-version=7.1` to connect to SQL Server.

When connecting to a database hosted with Microsoft Azure, FreeTDS must be compiled with OpenSSL, use the `--with-openssl` configure flag.

14.5 Generic Oracle backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserail: Yes
- Case: All lower
- DNSSEC: Yes (set `goracle-dnssec`)
- Disabled data: Yes
- Comments: Yes
- Module name: `goracle`
- Launch name: `goracle`

The Generic Oracle Backend is a *Generic SQL Backends*. 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          INTEGER NOT NULL,
  name        VARCHAR2(255) NOT NULL,
  master      VARCHAR2(128) DEFAULT NULL,
  last_check  INTEGER DEFAULT NULL,
  type        VARCHAR2(6) NOT NULL,
  notified_serial NUMBER(10,0) DEFAULT NULL,
  account     VARCHAR2(40) DEFAULT NULL,
  PRIMARY KEY (id)
);

CREATE SEQUENCE domains_id_sequence;
CREATE INDEX domains$name ON domains (name);

CREATE TABLE records (
  id          INTEGER NOT NULL,
  domain_id   INTEGER DEFAULT NULL REFERENCES domains (id) ON DELETE CASCADE,
  name        VARCHAR2(255) DEFAULT NULL,
  type        VARCHAR2(10) DEFAULT NULL,
  content     VARCHAR2(4000) DEFAULT NULL,
  ttl         INTEGER DEFAULT NULL,
  prio        INTEGER DEFAULT NULL,
  change_date INTEGER DEFAULT NULL,
  disabled    NUMBER(1,0) DEFAULT 0 NOT NULL,
  ordername   VARCHAR2(255) DEFAULT NULL,
  auth        NUMBER(1,0) DEFAULT 1 NOT NULL,
  PRIMARY KEY (id)
) pctfree 40;

CREATE SEQUENCE records_id_sequence;
CREATE INDEX records$name$type ON records (name, type);
CREATE INDEX records$domain_id ON records (domain_id);
CREATE INDEX records$recordorder ON records (domain_id, ordername);

CREATE TABLE supermasters (
  ip          VARCHAR2(64) NOT NULL,
  nameserver  VARCHAR2(255) NOT NULL,
```

(continues on next page)

(continued from previous page)

```

    account          VARCHAR2(40) NOT NULL,
    PRIMARY KEY (ip, nameserver)
);

CREATE TABLE comments (
    id                INTEGER NOT NULL,
    domain_id         INTEGER NOT NULL REFERENCES domains (id) ON DELETE CASCADE,
    name              VARCHAR2(255) NOT NULL,
    type              VARCHAR2(10) NOT NULL,
    modified_at       INTEGER NOT NULL,
    account            VARCHAR2(40) NOT NULL,
    "comment"         VARCHAR2(4000) NOT NULL
);
CREATE SEQUENCE comments_id_sequence;
CREATE INDEX comments$name$type ON comments (name, type);
CREATE INDEX comments$domain_id ON comments (domain_id);
CREATE INDEX comments$order ON comments (domain_id, modified_at);

CREATE TABLE domainmetadata (
    id                INTEGER NOT NULL,
    domain_id         INTEGER NOT NULL,
    kind              VARCHAR2(32),
    content            VARCHAR2(4000),
    PRIMARY KEY (id)
);
CREATE SEQUENCE domainmetadata_id_sequence;
CREATE INDEX domainmetadata$domain_id ON domainmetadata (domain_id);

CREATE TABLE cryptokeys (
    id                INTEGER NOT NULL,
    domain_id         INTEGER NOT NULL,
    flags             INTEGER NOT NULL,
    active            INTEGER NOT NULL,
    content            VARCHAR2(4000),
    PRIMARY KEY (id)
);
CREATE SEQUENCE cryptokeys_id_sequence;
CREATE INDEX cryptokeys$domain_id ON cryptokeys (domain_id);

CREATE TABLE tsigkeys (
    id                INTEGER NOT NULL,
    name              VARCHAR2(255),
    algorithm          VARCHAR2(50),
    secret             VARCHAR2(255),
    PRIMARY KEY (id)
);
CREATE SEQUENCE tsigkeys_id_sequence;
CREATE UNIQUE INDEX tsigkeys$namealgo ON tsigkeys (name, algorithm);

```

This schema contains all elements needed for master, slave and superslave operation.

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, 'example.
↪net', 'NATIVE');

```

14.5.1 Settings

`goracle-tnsname`

Which TNSNAME the Generic Oracle Backend should be connecting to. There are no `goracle-dbname`, `goracle-host` or `goracle-port` settings, their equivalent is in `/etc/tnsnames.ora`.

`goracle-dnssec`

Enable DNSSEC processing for this backend. Default=no.

14.5.2 Caveats

Password Expiry

When your password is about to expire, and logging into oracle warns about this, the Generic Oracle backend can no longer login, and will a OCILogin2 warning.

To work around this, either update the password in time or remove expiration from the account used.

14.6 Generic PostgreSQL backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserial: Yes
- Case: All lower
- DNSSEC: Yes (set `gppgsql-dnssec`)
- Disabled data: Yes
- Comments: Yes
- Module name: `gppgsql`
- Launch name: `gppgsql`

This PostgreSQL backend is based on the *Generic SQL Backends*. The default setup conforms to the schema at the bottom of this page, note that *zone2sql* with the `--gppgsql` flag also assumes this layout is in place.

This schema contains all elements needed for master, slave and superslave operation. For full migration notes, please see *Migration* docs.

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

14.6.1 Settings

`gppgsql-host`

Host (ip address) to connect to. If `gppgsql-host` begins with a slash, it specifies Unix-domain communication rather than TCP/IP communication; the value is the name of the directory in which the socket file is stored. Default: not set.

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.

`gpysql-port`

The port to connect to on *gpysql-host*. Default: not set.

`gpysql-dbname`

Name of the database to connect to. Default: not set.

`gpysql-user`

User to connect as. Default: not set.

`gpysql-password`

The password to for *gpysql-user*. Default: not set.

`gpysql-dnssec`

Enable DNSSEC processing for this backend. Default: no.

`gpysql-extra-connection-parameters`

Extra connection parameters to forward to postgres. If you want to pin a specific certificate for the connection you should set this to `sslmode=verify-full sslrootcert=<path-to-CA-cert>`. Accepted parameters are documented in the [PostgreSQL documentation](#). Default: "".

14.6.2 Default schema

```
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,
  CONSTRAINT c_lowercase_name CHECK ((name)::TEXT = LOWER((name)::TEXT))
);

CREATE UNIQUE INDEX name_index ON domains(name);

CREATE TABLE records (
  id          BIGSERIAL 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,
```

(continues on next page)

(continued from previous page)

```

prio                                INT DEFAULT NULL,
change_date                        INT DEFAULT NULL,
disabled                          BOOL DEFAULT 'f',
ordername                         VARCHAR(255),
auth                              BOOL DEFAULT 't',
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 INDEX recordorder ON records (domain_id, ordername text_pattern_ops);

CREATE TABLE supermasters (
    ip                                INET NOT NULL,
    nameserver                        VARCHAR(255) NOT NULL,
    account                          VARCHAR(40) NOT NULL,
    PRIMARY KEY(ip, nameserver)
);

CREATE TABLE comments (
    id                                SERIAL PRIMARY KEY,
    domain_id                        INT NOT NULL,
    name                            VARCHAR(255) NOT NULL,
    type                            VARCHAR(10) NOT NULL,
    modified_at                      INT NOT NULL,
    account                          VARCHAR(40) DEFAULT NULL,
    comment                          VARCHAR(65535) NOT 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 comments_domain_id_idx ON comments (domain_id);
CREATE INDEX comments_name_type_idx ON comments (name, type);
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);

CREATE TABLE domainmetadata (
    id                                SERIAL PRIMARY KEY,
    domain_id                        INT REFERENCES domains(id) ON DELETE CASCADE,
    kind                            VARCHAR(32),
    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
);

```

(continues on next page)

(continued from previous page)

```
CREATE INDEX domainidindex ON cryptokeys(domain_id);

CREATE TABLE tsigkeys (
  id          SERIAL PRIMARY KEY,
  name        VARCHAR(255),
  algorithm   VARCHAR(50),
  secret      VARCHAR(255),
  CONSTRAINT c_lowercase_name CHECK (((name)::TEXT = LOWER((name)::TEXT)))
);

CREATE UNIQUE INDEX namealgoindex ON tsigkeys(name, algorithm);
```

14.7 Generic SQLite 3 backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- DNSSEC: Yes
- Disabled data: Yes
- Comments: Yes
- Module name: gsqlite3
- Launch name: gsqlite3

Warning: When importing large amounts of data, be sure to run `analyze`; afterwards as SQLite3 has a tendency to use very sub-optimal indexes otherwise.

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 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 SQL Backends*.

SQLite exists in two incompatible versions, PowerDNS only supports version 3. To launch the backend, put `launch=gsqlite3` in the configuration.

14.7.1 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:

```
PRAGMA foreign_keys = 1;

CREATE TABLE domains (
  id          INTEGER PRIMARY KEY,
  name        VARCHAR(255) NOT NULL COLLATE NOCASE,
  master      VARCHAR(128) DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```

    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,
    disabled             BOOLEAN DEFAULT 0,
    ordername            VARCHAR(255),
    auth                 BOOL DEFAULT 1,
    FOREIGN KEY(domain_id) REFERENCES domains(id) ON DELETE CASCADE ON UPDATE CASCADE
);

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

CREATE TABLE supermasters (
    ip                  VARCHAR(64) NOT NULL,
    nameserver           VARCHAR(255) NOT NULL COLLATE NOCASE,
    account              VARCHAR(40) NOT NULL
);

CREATE UNIQUE INDEX ip_nameserver_pk ON supermasters(ip, nameserver);

CREATE TABLE comments (
    id                  INTEGER PRIMARY KEY,
    domain_id           INTEGER NOT NULL,
    name                VARCHAR(255) NOT NULL,
    type                VARCHAR(10) NOT NULL,
    modified_at          INT NOT NULL,
    account              VARCHAR(40) DEFAULT NULL,
    comment              VARCHAR(65535) NOT NULL,
    FOREIGN KEY(domain_id) REFERENCES domains(id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE INDEX comments_domain_id_index ON comments (domain_id);
CREATE INDEX comments_nametype_index ON comments (name, type);
CREATE INDEX comments_order_idx ON comments (domain_id, modified_at);

CREATE TABLE domainmetadata (
    id                  INTEGER PRIMARY KEY,
    domain_id            INT NOT NULL,
    kind                 VARCHAR(32) COLLATE NOCASE,
    content               TEXT,
    FOREIGN KEY(domain_id) REFERENCES domains(id) ON DELETE CASCADE ON UPDATE CASCADE

```

(continues on next page)

(continued from previous page)

```
);

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,
  FOREIGN KEY(domain_id) REFERENCES domains(id) ON DELETE CASCADE ON UPDATE CASCADE
);

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);
```

This schema contains all elements needed for master, slave and superslave operation.

Warning: It is not possible to replace the sqlite3 database file while PowerDNS is running. Specifically, using `rsync` to distribute sqlite3 databases does not work without stopping PowerDNS first and restarting it after the change.

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 --named-conf=/path/to/named.conf --gsqlite | sqlite3 powerdns.sqlite3`, but you can also use AXFR (or insert data manually).

To communicate with a SQLite database, use the `sqlite3` program, and feed it SQL.

14.7.2 Configuration Parameters

These are the configuration file parameters that are available for the gsqlite3 backend.

gsqlite3-database

Path to the SQLite3 database.

gsqlite3-pragma-synchronous

Set this to 0 for blazing speed.

gsqlite3-pragma-foreign-keys

Enable foreign key constraints.

gsqlite3-dnssec

Enable DNSSEC processing.

14.7.3 Using the SQLite backend

The last thing you need to do is telling PowerDNS to use the SQLite backend.

```
# in pdns.conf
launch=gsqlite3
gsqlite3-database=<path to your SQLite database>
```

Then you can start PowerDNS and it should notify you that a connection to the database was made.

14.7.4 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="gsqlite3"` to configure PowerDNS to use the SQLite backend. Compilation can then proceed as usual.

SQLite is included in most PowerDNS binary releases.

14.8 GeoIP backend

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- DNSSEC: Yes
- Disabled data: No
- Comments: No
- Module name: `geoip`
- Launch name: `geoip`

This backend allows visitors to be sent to a server closer 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. This backend can utilize EDNS Client Subnet extension for decision making, if provided in query and you have turned on *edns-subnet-processing*.

14.8.1 Prerequisites

To compile the backend, you need libyaml-cpp 0.5 or later and libgeoip.

You must have geoip database available. As of writing, on debian/ubuntu systems, you can use `apt-get install geoip-database` to get one, and the backend is configured to use the location where these files are installed as source. On other systems you might need to alter the `database-file` and `database-file6` attribute. If you don't need ipv4 or ipv6 support, set the respective setting to `""`. Leaving it unset leaves it pointing to default location, preventing the software from starting up.

Since v4.2.0 libgeoip is optional. You can use also libmaxminddb, but that is optional too. If no geo backend is provided, no geoip database based expansions can be used. Other expansions will work.

14.8.2 Configuration Parameters

These are the configuration file parameters that are available for the GeoIP backend. `geoip-zones-files` is the only thing you must set, if the defaults suite you.

`geoip-database-files`

Deprecated since version 4.2.0: This setting has been removed

Comma, tab or space separated list of files to open. You can use [geoip-cvs-to-dat](#) to generate your own.

For MMDB files, see <https://github.com/maxmind/getting-started-with-mmdb>

Since v4.2.0, database type is determined by file suffix, or you can use new syntax. New syntax is `[driver:]path[;options]`.

Currently supported options for dat driver (legacy libGeoIP):

- `mode=standard, memory, index or mmap`

Currently supported options for mmdb driver (libmaxminddb):

- `mode=mmap`
- `language=en` (which language to use)

Warning: This option has been changed since v4.2.0

`geoip-database-cache (before v4.2.0)`

Deprecated since version 4.2.0: This setting is removed

Specifies the kind of caching that is done on the database. This is one of “standard”, “memory”, “index” or “mmap”. These options map to the caching options described [here](#)

`geoip-zones-file`

Specifies the full path of the zone configuration file to use.

`geoip-dnssec-keydir`

Specifies the full path of a directory that will contain DNSSEC keys. This option enables DNSSEC on the backend. Keys can be created/managed with `pdnsutil`, and the backend stores these keys in files with key flags and active/disabled state encoded in the key filenames.

14.8.3 Zonefile format

Zone configuration file uses YAML syntax. Here is simple example. Note that the - before certain keys is part of the syntax.

```
domains:
- domain: geo.example.com
  ttl: 30
  records:
    geo.example.com:
      - soa: ns1.example.com hostmaster.example.com 2014090125 7200 3600 1209600 ↵
↵3600
      - ns:
          content: ns1.example.com
          ttl: 600
      - ns: ns2.example.com
      - mx: 10 mx.example.com
    fin.eu.service.geo.example.com:
      - a: 192.0.2.2
      - txt: hello world
      - aaaa: 2001:DB8::12:34DE:3
# this will result first record being handed out 30% of time
    swe.eu.service.geo.example.com:
      - a:
          content: 192.0.2.3
          weight: 50
      - a: 192.0.2.4
  services:
# syntax 1
    service.geo.example.com: '%co.%cn.service.geo.example.com'
# syntax 2
    service.geo.example.com: [ '%co.%cn.service.geo.example.com', '%cn.service.geo.
↵example.com']
# alternative syntax
  services:
    service.geo.example.com:
      default: [ '%co.%cn.service.geo.example.com', '%cn.service.geo.example.com' ]
      10.0.0.0/8: 'internal.service.geo.example.com'
```

Keys explained

- **domains:** Mandatory root key. All configuration is below this
- **domain:** Defines a domain. You need ttl, records, services under this.
- **ttl:** TTL value for all records
- **records:** Put fully qualified name as subkey, under which you must define at least soa: key. Note that this is an array of records, so - is needed for the values.
- **services:** Defines one or more services for querying.
- From 4.2.0, you can also use %lat, %lon, %loc to expand for geographic location, if available in backend. %loc in particular can be safely used with LOC record type.
- From 4.2.0, you can also use %ip4 and %ip6 that will expand to the IP address when AFI matches, and empty otherwise. Can be particularly used with A and AAAA record types.
- From 4.1.0, you can also use %cc = 2 letter country code
- From 4.0.0, you can also use %as = ASn, %ip = Remote IP
- From 4.0.0, you can also use additional specifiers. These are %hh = hour, %dd = day, %mo = month, %mos = month as short string, %wd = weekday (as number), %wds weekday as short string.
- From 4.0.0, scopeMask is set to most specific value, in case of date/time modifiers it will be 32 or 128, but with the others it is set to what geoip says it used for matching.

- From 4.0.0, You can add per-network overrides for format, they will be formatted with the same placeholders as default. Default is short-hand for adding 0.0.0.0/0 and ::/0. Default is default when only string is given for service name.
- From 4.0.0, You can use array to specify return values, works only if you have those records specified. It matches the format results to your records, and if it finds match that is used. Otherwise the last is returned.
- From 4.0.0, You can apply all the attributes for the content of static records too.
- From 4.0.0, You can use record attributes to set TTL.
- From 4.0.0, You can use record attributes to define weight. If this is given, only one record is chosen randomly based on the weight. **DO NOT** mix record types for these. It will not work. **PROBABILITY** is calculated by summing up the weights and dividing each weight with the sum. **WARNING:** If you use ip or time/date specifiers, caching will be disabled for that RR completely. That means, if you have a

something.example.com: - a: 1.2.3.4 - txt: "your ip is %ip"

then caching will not happen for any records of something.example.com. If you need to use TXT for debugging, make sure you use dedicated name for it.

Since v4.1.0 you can mix service and static records to produce the sum of these records, including apex record.

Format explained

Following placeholders are supported, and support subnet caching with EDNS. - %%%: % - %co: With legacy GeoIP database only expands to three letter country name,

with MMDB and others this will expand into ISO3166 country code.

- %cc: ISO3166 country code.
- %cn: ISO3166 continent code.
- %af: v4 or v6.
- %re: Region code
- %na: AS organization name (spaces are converted to _)
- %as: AS number
- %ci: City name
- %loc: LOC record style expansion of location
- %lat: Decimal degree latitude
- %lon: Decimal degree longitude

Following placeholders disable caching completely. - %yy: Year - %mos: Month name - %mo: Month - %wds: Weekday name - %wd: Weekday - %dd: Year day - %hh: Hour - %ip: IP address - %ip4: IPv4 address - %ip6: IPv6 address

Warning: Before 4.2.0 if record expanded to empty value it could cause SERVFAIL. Since 4.2.0 such expansions for non-TXT record types are not included in response.

Warning: If the record which a service points to exists under "records" then it is returned as a direct answer. If it does not exist under "records" then it is returned as a CNAME.

Warning: If your services match wildcard records in your zone file then these will be returned as CNAMEs. This will only be an issue if you are trying to use a service record at the apex of your domain where you need other record types to be present (such as NS and SOA records.) Per [RFC 2181](#), CNAME records cannot appear in the same label as NS or SOA records.

14.9 LDAP backend

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- Autoserail: No
- DNSSEC: No
- Disabled data: No
- Comments: No
- Module name: ldap
- Launch name: ldap

14.9.1 Introduction

As of PowerDNS Authoritative Server 4.0.0, the LDAP backend is fully supported.

The original author for this module is Norbert Sendetzky. This page is based on the content from his [LDAPbackend wiki section](#) as copied in February 2016, and edited from there.

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.

Rationale

The LDAP backend enables PowerDNS to retrieve DNS information from any standard compliant LDAP server. This is extremely handy if information about hosts is already stored in an LDAP tree.

Schemas

The schema is based on the ‘uninett’ dnszone schema, with a few types added by number as designed in that schema:

```
# A schema for storing DNS zones in LDAP
#
# ORDERING is not necessary, and some servers don't support
# integerOrderingMatch. Omit or change if you like

attributetype ( 1.3.6.1.4.1.2428.20.0.0 NAME 'dnSTTL'
    DESC 'An integer denoting time to live'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
```

(continues on next page)

(continued from previous page)

```
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

attributetype ( 1.3.6.1.4.1.2428.20.0.1 NAME 'dNSClass'
  DESC 'The class of a resource record'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.11 NAME 'wKSRecord'
  DESC 'a well known service description, RFC 1035'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.12 NAME 'pTRRecord'
  DESC 'domain name pointer, RFC 1035'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.13 NAME 'hInfoRecord'
  DESC 'host information, RFC 1035'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.14 NAME 'mInfoRecord'
  DESC 'mailbox or mail list information, RFC 1035'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.16 NAME 'tXTRecord'
  DESC 'text string, RFC 1035'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.17 NAME 'rPRecord'
  DESC 'for Responsible Person, RFC 1183'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.18 NAME 'aFSDBRecord'
  DESC 'for AFS Data Base location, RFC 1183'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.24 NAME 'SigRecord'
  DESC 'Signature, RFC 2535'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.25 NAME 'KeyRecord'
  DESC 'Key, RFC 2535'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

(continues on next page)

(continued from previous page)

```
attributetype ( 1.3.6.1.4.1.2428.20.1.27 NAME 'gPosRecord'
    DESC 'Geographical Position, RFC 1712'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.28 NAME 'aAAARecord'
    DESC 'IPv6 address, RFC 1886'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.29 NAME 'LocRecord'
    DESC 'Location, RFC 1876'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.30 NAME 'nXTRecord'
    DESC 'non-existant, RFC 2535'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.33 NAME 'sRVRecord'
    DESC 'service location, RFC 2782'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.35 NAME 'nAPTRRecord'
    DESC 'Naming Authority Pointer, RFC 2915'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.36 NAME 'kXRecord'
    DESC 'Key Exchange Delegation, RFC 2230'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.37 NAME 'certRecord'
    DESC 'certificate, RFC 2538'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.38 NAME 'a6Record'
    DESC 'A6 Record Type, RFC 2874'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.39 NAME 'dNameRecord'
    DESC 'Non-Terminal DNS Name Redirection, RFC 2672'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.42 NAME 'aPLRecord'
```

(continues on next page)

(continued from previous page)

```
DESC 'Lists of Address Prefixes, RFC 3123'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.43 NAME 'dSRecord'
DESC 'Delegation Signer, RFC 3658'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.44 NAME 'sSHFPRecord'
DESC 'SSH Key Fingerprint, RFC 4255'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.45 NAME 'iPSecKeyRecord'
DESC 'SSH Key Fingerprint, RFC 4025'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.46 NAME 'rRSIGRecord'
DESC 'RRSIG, RFC 3755'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.47 NAME 'nSECRecord'
DESC 'NSEC, RFC 3755'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.48 NAME 'dNSKeyRecord'
DESC 'DNSKEY, RFC 3755'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.49 NAME 'dHCIDRecord'
DESC 'DHCID, RFC 4701'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.50 NAME 'nSEC3Record'
DESC 'NSEC record version 3, RFC 5155'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.51 NAME 'nSEC3PARAMRecord'
DESC 'NSEC3 parameters, RFC 5155'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.52 NAME 'tLSARecord'
DESC 'TLSA certificate association, RFC 6698'
```

(continues on next page)

(continued from previous page)

```
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.59 NAME 'cDSRecord'
    DESC 'Child DS, RFC7344'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.60 NAME 'cDNSKeyRecord'
    DESC 'DNSKEY(s) the Child wants reflected in DS, RFC7344'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.61 NAME 'openPGPKeyRecord'
    DESC 'OpenPGP Key, RFC7929'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.99 NAME 'sPFRecord'
    DESC 'Sender Policy Framework, RFC 4408'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.108 NAME 'EUI48Record'
    DESC 'EUI-48 address, RFC7043'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.109 NAME 'EUI64Record'
    DESC 'EUI-64 address, RFC7043'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.249 NAME 'tKeyRecord'
    DESC 'Transaction Key, RFC2930'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.256 NAME 'uRIRRecord'
    DESC 'URI, RFC7553'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.257 NAME 'cAARRecord'
    DESC 'Certification Authority Restriction, RFC6844'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.32769 NAME 'dLVRecord'
    DESC 'DNSSEC Lookaside Validation, RFC4431'
    EQUALITY caseIgnoreIA5Match
```

(continues on next page)

(continued from previous page)

```

SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.65226 NAME 'TYPE65226Record'
DESC ''
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.65534 NAME 'TYPE65534Record'
DESC ''
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

objectclass ( 1.3.6.1.4.1.2428.20.2 NAME 'dnsDomain2'
SUP 'dnsDomain' STRUCTURAL
MAY ( DNSTTL $ DNSClass $ WKSRecord $ PTRRecord $
HINFORecord $ MINFORecord $ TXTRecord $ RPRecord $
AFSDBRecord $ SIGRecord $ KEYRecord $ GPOSRecord $
AAAARecord $ LOCRecord $ NXTRecord $ SRVRecord $
NAPTRRecord $ KXRecord $ CERTRecord $ A6Record $
DNAMERecord $ APLRecord $ DSRecord $ SSHFPRecord $
IPSECKEYRecord $ RRSIGRecord $ NSECRecord $
DNSKEYRecord $ DHCIDRecord $ NSEC3Record $ NSEC3PARAMRecord $
TLSARecord $ CDSRecord $ CDNSKEYRecord $ OPENPGPKEYRecord $
SPFRecord $ EUI48Record $ EUI64Record $ TKEYRecord $
URIRecord $ CAAREcord $ DLVRecord $ TYPE65226Record $
TYPE65534Record
) )

```

The LDAP `dnsdomain2` schema contains the additional object descriptions which are required by the LDAP server to check the validity of entries when they are added. Please consult the documentation of the LDAP server to find out how to add this schema to the server.

14.9.2 Installation

The LDAP backend can be compiled by adding `ldap` to either the `--with-modules` or `--with-dynmodules` configure options.

When using packages, the `pdns-backend-ldap` package should be installed.

14.9.3 Configuration options

There are a few options through the LDAP DNS backend can be configured. Add them to the `pdns.conf` file.

To launch the `ldap` backend:

```
launch=ldap
```

ldap-host

(default `"ldap://127.0.0.1:389"`): The values assigned to this parameter can be LDAP URIs (e.g. `ldap://127.0.0.1/` or `ldaps://127.0.0.1/`) describing the connection to the LDAP server. There can be multiple LDAP URIs specified for load balancing and high availability if they are separated by spaces. In case the used LDAP client library doesn't support LDAP URIs as connection parameter, use plain host names or IP addresses instead (both may optionally be followed by a colon and the port).

ldap-starttls

(default “no”) : Use TLS encrypted connections to the LDAP server. This is only allowed if ldap-host is a [ldap://](#) URI or a host name / IP address.

ldap-timeout

(default: “5”) : The number of seconds to wait for LDAP operations to complete.

ldap-reconnect-attempts

(default: “5”) : The number of attempts to make to re-establish a lost connection to the LDAP server.

ldap-authmethod

(default: “simple”) : How to authenticate to the LDAP server. Actually only two methods are supported: “simple”, which uses the classical DN / password, or “gssapi”, which requires a Kerberos keytab.

ldap-binddn

(default “”) : Path to the object to authenticate against. Should only be used, if the LDAP server doesn’t support anonymous binds and with the “simple” authmethod.

ldap-secret

(default “”) : Password for authentication against the object specified by ldap-binddn. Only used when “authmethod” is “simple”.

ldap-krb5-keytab

(default: “”) : Full path to the keytab file to use to authenticate. This is only used when “authmethod” is set to “gssapi”. The keytab must, ideally, contain only one principal (or to put it otherwise, only the first principal found in the keytab will be used).

ldap-krb5-ccache

(default: “”) : Full path to the Kerberos credential cache file to use. Actually only files are supported, and the “FILE:” prefix must not be set. The PowerDNS process must be able to write to this file and it *must* be the only one able to read it.

ldap-basedn

(default “”) : The PowerDNS LDAP DNS backend searches below this path for objects containing the specified DNS information. The retrieval of attributes is limited to this subtree. This option must be set to the path according to the layout of your LDAP tree, e.g. ou=hosts,o=linuxnetworks,c=de is the DN to my objects containing the DNS information.

ldap-method

(default “simple”) :

- **simple**: Search the requested domain by comparing the associatedDomain attributes with the domain string in the question.
- **tree**: Search entires by translating the domain string into a LDAP dn. Your LDAP tree must be designed in the same way as the DNS LDAP tree. The question for “myhost.linuxnetworks.de” would translate into “dc=myhost,dc=linuxnetworks,dc=de,ou=hosts=...” and the entry where this dn points to would be evaluated for dns records.
- **strict**: Like simple, but generates PTR records from aRecords or aAAARecords. Using “strict”, zone transfers for reverse zones are not possible.

ldap-filter-axfr

(default “(:target:)”) : LDAP filter for limiting AXFR results (zone transfers), e.g. (&(:target:)(active=yes)) for returning only entries whose attribute “active” is set to “yes”.

ldap-filter-lookup

(default “(:target:)”) : LDAP filter for limiting IP or name lookups, e.g. (&(:target:)(active=yes)) for returning only entries whose attribute “active” is set to “yes”.

14.9.4 Master Mode

Schema update

First off adding master support to the LDAP backend needs a schema update. This is required as some metadata must be stored by PowerDNS, such as the last successful transfer to slaves. The new schema is available in schema/pdns-domaininfo.schema.

Once the schema is loaded the zones for which you want to be a master must be modified. The dn of the SOA record *must* have the object class PdnsDomain, and thus the PdnsDomainId attribute. This attribute is an integer that *must* be unique across all zones served by the backend. Furthermore the PdnsDomainType must be equal to ‘master’ (lower case).

Example

Here is an example LDIF of a zone that’s ready for master operation (assuming the ‘tree’ style):

```
dn: dc=example,dc=com,ou=dns,dc=mycompany,dc=com
objectClass: top
objectClass: domainRelatedObject
objectClass: dNSDomain2
objectClass: PdnsDomain
dc: example
associatedDomain: example.com
nsRecord: ns1.example.com
soaRecord: ns1.example.com. hostmaster.example.com. 2013031101 1800 600 1209600 600
mXRecord: 10 mx1.example.com
PdnsDomainId: 1
PdnsDomainType: master
PdnsDomainMaster: 192.168.0.2
```

You should have one attribute PdnsDomainMaster per master serving this zone.

14.9.5 Example

Tree design

The DNS LDAP tree should be designed carefully to prevent mistakes, which are hard to correct afterwards. The best solution is to create a subtree for all host entries which will contain the DNS records. This can be done the simple way or in a tree style.

DN of a simple style example record (e.g. myhost.example.com):

```
dn:dc=myhost,dc=example,ou=hosts,...
```

DN of a tree style example record (e.g. myhost.test.example.com):

```
dn:dc=myhost,dc=test,dc=example,dc=com,ou=hosts,...
```

Basic objects

Each domain (or zone for BIND users) must include one object containing a SOA (Start Of Authority) record. This requirement applies to both forward and reverse zones. This object can also contain the attribute for a MX (Mail eXchange) and one or more NS (Name Server) records. These attributes allow one or more values, e.g. for a backup mail or name server:

```
dn:dc=example,ou=hosts,o=example,c=com
objectclass:top
objectclass:dcobject
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:example
soarecord:ns.example.com me@example.com 1 1800 3600 86400 7200
nsrecord:ns.example.com
mxrecord:10 mail.example.com
mxrecord:20 mail2.example.com
associateddomain:example.com
```

A simple mapping between name and IP address can be specified by an object containing an arecord and an associateddomain.

```
dn:dc=server,dc=example,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:server
arecord:10.1.0.1
arecord:192.168.0.1
associateddomain:server.example.com
```

Be aware of the fact that these examples work if `ldap-method` is `simple` or `strict`. For tree mode, all DN's will have to be modified according to the algorithm described in the section above.

Wildcards

Wild-card domains are possible by using the asterisk in the `associatedDomain` value like it is used in the bind zone files. The “dc” attribute can be set to any value in simple or strict mode - this doesn't matter.

```
dn:dc=any,dc=example,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:any
```

(continues on next page)

(continued from previous page)

```
arecord:192.168.0.1
associateddomain:*.example.com
```

In tree mode wild-card entries has to look like this instead:

```
dn:dc=*,dc=example,dc=de,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:*
arecord:192.168.0.1
associateddomain:*.example.com
```

Aliases

Aliases for an existing DNS object have to be defined in a separate LDAP object. One object should be create per alias (this is a must in tree mode) or add all aliases (as values of `associateddomain`) to one object. The only thing which is not allowed is to create loops by using the same name in `associateddomain` and in `cnamerecord`.

```
dn:dc=server-aliases,dc=example,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:server-aliases
cnamerecord:server.example.com
associateddomain:proxy.example.com
associateddomain:mail2.example.com
associateddomain:ns.example.com
```

Aliases are optional. All alias domains can also be added to the `associateddomain` attribute. The only difference is that these additional domains aren't recognized as aliases anymore, but instead as a normal `arecord`:

```
dn:dc=server,dc=example,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain
objectclass:domainrelatedobject
dc:server
arecord:10.1.0.1
associateddomain:server.example.com
associateddomain:proxy.example.com
associateddomain:mail2.example.com
associateddomain:ns.example.com
```

Reverse lookups

Currently there are two options: Set `ldap-method` to `strict` to have the code automatically derive PTR records from A and AAAA records in the tree. Or, in `simple` and `tree` modes, create additional objects explicitly mapping each address to a PTR record.

For `strict` or `simple` modes, first create an object with an SOA record for the reverse-lookup zone(s) corresponding to the A and AAAA records that will be served:

```
dn:dc=1.10.in-addr.arpa,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain2
objectclass:domainrelatedobject
dc:1.10.in-addr.arpa
```

(continues on next page)

(continued from previous page)

```
soarecord:ns.example.com me@example.com 1 1800 3600 86400 7200
nsrecord:ns.example.com
associateddomain:1.10.in-addr.arpa
```

In strict mode, no other objects are required – reverse queries that correspond to an arecord or aaaarecord of an existing object will be automagically serviced using the associateddomain entry of that object.

In simple mode, you must then create objects for each reverse mapping:

```
dn:dc=1.0,dc=1.10.in-addr.arpa,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain2
objectclass:domainrelatedobject
dc:1.0
ptrrecord:server.example.com
associateddomain:1.0.1.10.in-addr.arpa
```

Tree mode requires each component to be a dc element of its own:

```
dn:dc=1,dc=0,dc=1,dc=10,dc=in-addr,dc=arpa,ou=hosts,o=example,c=de
objectclass:top
objectclass:dnsdomain2
objectclass:domainrelatedobject
dc:1
ptrrecord:server.example.com
associateddomain:1.0.1.10.in-addr.arpa
```

To use this kind of record, add the dnsdomain2 schema to the configuration of ther LDAP server.

CAUTION: `ldap-method=strict` can not be used if zone transfers (AXFR) are needed to other name servers. Distributing zones can only be done directly via LDAP replication in this case, because for a full zone transfer the reverse records are missing.

14.9.6 Migration

BIND zone files

There is a small utility in the PowerDNS distribution available called *zone2ldap*, which can convert zone files used by BIND to the ldif format. Ldif is a text file format containing information about LDAP objects and can be read by every standard compliant LDAP server. *zone2ldap* needs the BIND `named.conf` (usually located in `/etc`) as input and writes the dns record entries in ldif format to stdout:

```
zone2ldap
--basedn=YOUR_BASE_DN \
--named-conf=PATH_TO_NAMED_CONF \
--resume > zones.ldif
```

Alternatively *zone2ldap* can be used to convert only single zone files instead all zones:

```
zone2ldap
--basedn=YOUR_BASE_DN \
--zone-file=PATH_TO_ZONE_FILE \
--zone-name=NAME_OF_ZONE \
--resume > zone.ldif
```

See *its manpage* for a complete list of options.

Bind LDAP backend

When coming from the [Bind LDAP sdb backend](#), the records can be kept in the LDAP tree also for the PowerDNS LDAP backend. The schemas both backends utilize is almost the same except for one important thing: Domains for PowerDNS are stored in the attribute “associatedDomain” whereas Bind stores them split in “relativeDomainName” and “zoneName”.

There is a [migration script](#) which creates a file in LDIF format with the necessary LDAP updates including the “associatedDomain” and “dc” attributes. The utility is executed on the command line by:

```
./bind2pdns-ldap
--host=HOSTNAME_OR_IP \
--basedn=YOUR_BASE_DN \
--binddn=ADMIN_DN > update.ldif
```

The parameter “host” and “basedn” are mandatory, “binddn” is optional. If “binddn” is given, the script will prompt for a password, otherwise an anonymous bind is executed. The updates in LDIF format are written to stdout and can be redirected to a file.

The script requires Perl and the Perl Net::LDAP module and can be downloaded [here](#).

Updating the entries in the LDAP tree requires to make the dnsdomain2 schema known to the LDAP server. Unfortunately, both schemas (dnsdomain2 and dnszone) share the same record types and use the same OIDs so the LDAP server can’t use both schemas at the same time. The solution is to add the [dnsdomain2 schema](#) and replace the dnszone schema by the [dnszone-migrate schema](#). After restarting the LDAP server attributes from both schemas can be used and updating the objects in the LDAP tree using the LDIF file generated from bind2pdns-ldap will work without errors.

Other name server

The easiest way for migrating DNS records is to use the output of a zone transfer (AXFR). Save the output of the dig program provided by bind into a file and call zone2ldap with the file name as option to the --zone-file parameter. This will generate the appropriate ldif file, which can be imported into the LDAP tree. The bash script except below automates this:

```
DNSSERVER=127.0.0.1
DOMAINS="example.com 10.10.in-addr.arpa"

for DOMAIN in $DOMAINS; do
    dig @$DNSSERVER $DOMAIN AXFR> $DOMAIN.zone;
    zone2ldap --zone-name=$DOMAIN --zone-file=$DOMAIN.zone> $DOMAIN.ldif;
done
```

14.9.7 Optimization

LDAP indices

To improve performance, the LDAP server can maintain indices on certain attributes. This leads to much faster searches for these type of attributes.

The LDAP DNS backend mainly searches for values in associatedDomain, so maintaining an index (pres,eq,sub) on this attribute is a big performance improvement:

```
indexassociatedDomain pres,eq,sub
```

Furthermore, if ldap-method=strict is set, it uses the aRecord and aAAARRecord attribute for reverse mapping of IP addresses to names. To maintain an index (pres,eq) on these attributes also improves performance of the LDAP server:

```
indexaAAAARecord pres,eq
indexaRecord pres,eq
```

All other attributes than `associatedDomain`, `aRecord` or `aAAAARecord` are only read if the object matches the specified criteria. Thus, maintaining an index on these attributes is useless.

If the DNS-entries were added before adding these statements to `slapd.conf`, the LDAP server will have to be stopped and `slapindex` should be used on the command line. This will generate the indices for already existing attributes.

dNSTTL attribute

Converting the string in the `dNSTTL` attribute to an integer is a time consuming task. If no separate TTL value for each entry is required, use the `default-ttl` parameter in `pdns.conf` instead. This will gain a 7% improvement in performance for entries that aren't cached. A `dNSTTL` attribute can still be added to entries that should have a different TTL than the default TTL.

Access method

The method of accessing the entries in the directory affects the performance too. By default, the “simple” method is used search for entries by using their `associatedDomain` attribute. Alternatively, the “tree” method can be used, whereby the search is done along the directory tree, e.g. “host.example.com” is translated into “dc=host,dc=example,dc=com,...”. This requires the LDAP DNS subtree layout to be 1:1 to the DNS tree, this will gain an additional 7% performance improvement.

14.9.8 Troubleshooting

No reverse zone transfer

The LDAP tree must contain a separate subtree of PTR records (e.g. for 1.1.10.10.in-addr.arpa) and `ldap-method` can't be set to “strict”.

IPv6 reverse lookup doesn't work in strict mode

For automatically generated reverse IPv6 records the `aAAAARecord` entries must follow two restrictions: They have to be fully expanded (“FFFF::1” is not allowed and it must be “FFFF:0:0:0:0:0:1” instead) and they must not contain leading zeros, e.g. an entry containing “002A” is incorrect - use “2A” without zeros instead. These restrictions are due to the fact that LDAP DNS AAAA entries are pure text and doesn't allow searching by wild-cards.

14.9.9 Future

DNS notification support

As soon as the LDAP server implementations begin to provide the features of the LDAP client update protocol (LCUP, [RFC 3928](#)), it will be possible to support the DNS notification feature for the LDAP DNS backend in case a record in the LDAP directory was changed.

SASL support

Support for more authentication methods would be handy. Anyone interested may [contribute](#).

14.10 Lua Backend

- Native: Yes
- Master: Yes
- Slave: No
- Superslave: No
- Autoserail: No
- DNSSEC: Yes
- Disabled data: Yes
- Comments: Yes
- Module name: lua
- Launch name: lua

The main author for this module is Fredrik Danerklint.

This backend is just a “glue” between PowerDNS and your own Lua application.

What this means is that you can not have a working setup that can serve you dns-questions directly from start. What you need to do is to program your own backend completely in Lua! Which database server to use etc is now up to you!

What you have here is the possibility to make your own “dns-server” without the knowledge of programming in c/c++.

There is one thing that needs to be said. Remember that each thread PowerDNS launches of this backend is completely different so they cannot share information between each other!

You will need some kind of a database that can be shared for this.

All the functionnames that PowerDNS accept for a backend should be the same in your Lua script, in lowercase. Also, the parameters should be in the same order. Where there is a structure in c/c++ there is a table in the Lua backend. This is also true for return values. A few functions expect that you return a table in a table.

14.10.1 New functions

There is a couple of new functions for you to use in Lua:

`logger(log_facility, "your", "messages")`

All these `log_facilities` is available:

- `log_all`
- `log_ntlog`
- `log_alert`
- `log_critical`
- `log_error`
- `log_warning`
- `log_notice,`
- `log_info`
- `log_debug`
- `log_none`

dnspacket ()

This will give you back three parameters with `remote_ip`, `remote_port` and `local_ip` in that order.
Can only be used in the functions `list ()` and `getsoa ()`.

getarg ("PARAMETER")

This one tries to get the value of the name "`lua-PARAMETER`" from the `pdns.conf` file.

mustdo ("PARAMETER")

This is the same as `getarg ()` <#getarg> but return a boolean instead of a string.
You also have all the different QTypes in a table called 'QTypes'.

14.10.2 What has been tested

The only functionality of the minimal functions except zone-transfer has been tested.

In the included `powerdns-luabackend.lua` file there is a example of how this can be done. Note that this is more or less a static example since there is no possibility for each thread to know when something has changed.

However, you can run `pdns_control reload` and it should reload the whole thing from scratch (does not work for the moment, PowerDNS only calls two thread with the reload command - not all of them).

14.10.3 What you will find under the test directory

The following script can be used to test the server:

This will yield the following result:

```
$dig any www.test.com @127.0.0.1 -p5300 +multiline
; <<>> DiG 9.7.3 <<>> any www.test.com @127.0.0.1 -p5300 +multiline
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 1001
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.test.com.          IN ANY

;; ANSWER SECTION:
www.test.com.          120 IN CNAME host.test.com.
host.test.com.         120 IN A 10.11.12.13
host.test.com.         120 IN AAAA 1:2:3:4:5:6:7:8

;; Query time: 1 msec
;; SERVER: 127.0.0.1#5300(127.0.0.1)
;; WHEN: Thu Jun  2 22:19:56 2011
;; MSG SIZE  rcvd: 93
```

14.10.4 Parameters

lua-filename

Path to your lua script, '`powerdns-luabackend.lua`' by default.

`lua-logging-query`

Log queries. default is 'no'.

`lua-f_FUNCTION=NEWFUNCTION`

You can also override all the default functionsnames for the luafunctions if you want. For example:

```
lua-f_lookup = mynewfunction
```

will call the function `mynewfunction` for the lookup-routine.

If you want your own configuration parameters you can have that too. Just call the function `getarg("PARAMETER")` and it will return the value of `lua-PARAMETER`. For boolean you use the function `mustdo("PARAMETER")`.

Your own error function in lua

You can have an error function in Lua when Lua gives back a error.

First make your error function then you put this in `pdns.conf`:

```
lua-f_exec_error = YOUR_METHOD
```

14.10.5 DNSSEC

You can have full dnssec support in our Lua application. You should note the following regarding this:

You don't have to implement the function 'updateDNSSECOrderAndAuth' since the default code will work correctly for you via the backend itself.

The functions `activateDomainKey` and `deactivateDomainKey` can be implemented via a new function called `updateDomainKey`, which has three parameters (the other two has only two parameters) where the third is a boolean which is true or false depending on which function that was called from the beginning.

14.10.6 Information for logging

If you have the parameter `query-logging` or `lua-logging-query` set to `true/yes/on`, then you will see what is happening in each function when PowerDNS calls them.

This can, hopefully, help you with some debugging if you run into some kind of trouble with your Lua application.

14.11 MyDNS Backend

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- Autoserail: No
- Case: Depends
- DNSSEC: No
- Disabled data: No
- Comments: No

- Module name: `mydns`
- Launch name: `mydns`

The MyDNS backend makes PowerDNS a drop-in replacement for the [MyDNS](#) nameserver, as it uses the same database schema.

14.11.1 Configuration Parameters

`mydns-host`

Database host to connect to.

`mydns-port`

Port on the database server to connect to.

`mydns-dbname`

Name of the database to connect to, “mydns” by default.

`mydns-user`

User for the database, “powerdns” by default.

`mydns-password`

The user password.

`mydns-socket`

Unix socket to connect to the database.

`mydns-rr-table`

Name of the resource record table in the database, “rr” by default.

`mydns-soa-table`

Name of the SOA table in the database, “soa” by default.

`mydns-soa-where`

Additional WHERE clause for SOA, default is “1 = 1”.

`mydns-rr-where`

Additional WHERE clause for resource records, default is “1 = 1”.

`mydns-soa-active`

Use the active column in the SOA table, “yes” by default.

`mydns-rr-active`

Use the active column in the resource record table, “yes” by default.

`mydns-use-minimal-ttl`

Setting this to ‘yes’ will make the backend behave like MyDNS on the TTL values. Setting it to ‘no’ will make it ignore the minimal-ttl of the zone. The default is “yes”.

14.11.2 Migrating from MyDNS to another SQL backend

To use one of the *generic SQL backend*, like the *Postgresql* or *MySQL* backends, the data can be migrated using the *Backend to Backend* migration guide.

14.12 OpenDBX Backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserail: Yes
- DNSSEC: No
- Disabled data: No
- Comments: No
- Module name: `opendbx`
- Launch name: `opendbx`

The OpenDBX backend allows the authoritative server to connect to any backend supported by [OpenDBX](#).

This document contains a subset of the [full documentation](#) supplied by the author Norbert Sendetzky . This module is fully supported (and tested) by PowerDNS.

The OpenDBX backend has a mechanism to connect different database servers for read and write actions.

The domains table for the `opendbx` backend has a “status” column, when set to “A”, the domain is considered active and is actually served.

14.12.1 Settings

`opendbx-backend`

Name of the backend used to connect to the database server. Currently `mysql`, `pgsql`, `sqlite`, `sqlite3` and `sybase` are available. Default=`mysql`.

opendbx-host-read

One or more host names or IP addresses of the database servers. These hosts will be used for retrieving the records via SELECT queries. Default=127.0.0.1

opendbx-host-write

One or more host names or IP addresses of the database servers. These hosts will be used for INSERT/UPDATE statements (mostly used by zonetransfers). Default=127.0.0.1

opendbx-port

TCP/IP port number where the database server is listening to. Most databases will use their default port if you leave this empty.

opendbx-database

The database name where all domain and record entries are stored. Default=powerdns

opendbx-username

Name of the user send to the DBMS for authentication. Default=powerdns.

opendbx-password

Clear text password for authentication in combination with the username.

14.12.2 Queries

As with the *Generic SQL Backends*, queries are configurable. Note: If you change one of the SELECT statements must not change the order of the retrieved columns! To get the default queries, run `pdns_server --no-config --launch=opendbx --config`. The following queries are configurable:

- `opendbx-sql-list`: Select records which will be returned to clients asking for zone transfers (AXFR).
- `opendbx-sql-lookup`: Retrieve DNS records by name.
- `opendbx-sql-lookupid`: Retrieve DNS records by id and name.
- `opendbx-sql-lookuptype`: Retrieve DNS records by name and type.
- `opendbx-sql-lookuptypeid`: Retrieve DNS records by id, name and type.
- `opendbx-sql-lookupsoa`: Retrieve SOA record for domain.
- `opendbx-sql-zonedelete`: Delete all records from zone before inserting new ones via AXFR.
- `opendbx-sql-zoneinfo`: Get stored information about a domain.
- `opendbx-sql-transactbegin`: Start transaction before updating a zone via AXFR.
- `opendbx-sql-transactend`: Commit transaction after updating a zone via AXFR.
- `opendbx-sql-transactabort`: Undo changes if an error occurred while updating a zone via AXFR.
- `opendbx-sql-insert-slave`: Adds a new zone from the authoritative DNS server which is currently retrieved via AXFR.
- `opendbx-sql-insert-record`: Adds new records of a zone form the authoritative DNS server which are currently retrieved via AXFR.

- `opendbx-sql-update-serial`: Set zone serial to value of last update.
- `opendbx-sql-update-lastcheck`: Set time of last zone check.
- `opendbx-sql-master`: Get master record for zone.
- `opendbx-sql-supermaster`: Get supermaster info.
- `opendbx-sql-infoslaves`: Get all unfresh slaves.
- `opendbx-sql-infomasters`: Get all updates masters.

14.12.3 Database schemas and information

Mysql

The file below also contains trigger definitions which are necessary for *Autoserial* support, but they are only available in MySQL 5 and later. If you are still using MySQL 4.x and don't want to utilize the automatically generated zone serials, you can safely remove the "CREATE TRIGGER" statements from the file before creating the database tables.

```
SET SESSION sql_mode='ANSI';

CREATE TABLE "domains" (
  "id" INTEGER NOT NULL AUTO_INCREMENT,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "master" VARCHAR(40) NOT NULL DEFAULT '',
  "account" VARCHAR(40) NOT NULL DEFAULT '',
  "last_check" INTEGER DEFAULT NULL,
  "notified_serial" INTEGER DEFAULT NULL,
  "auto_serial" INTEGER NOT NULL DEFAULT 0,
  "status" CHAR(1) NOT NULL DEFAULT 'A',
  CONSTRAINT "pdns_pk_domains_id"
    PRIMARY KEY ("id"),
  CONSTRAINT "pdns_unq_domains_name"
    UNIQUE ("name")
) type=InnoDB;

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
  "id" INTEGER NOT NULL AUTO_INCREMENT,
  "domain_id" INTEGER NOT NULL,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "ttl" INTEGER DEFAULT NULL,
  "prio" INTEGER DEFAULT NULL,
  "content" VARCHAR(255) NOT NULL,
  CONSTRAINT "pdns_pk_records_id"
    PRIMARY KEY ("id"),
  CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
      REFERENCES "domains" ("id")
      ON UPDATE CASCADE
      ON DELETE CASCADE
) type=InnoDB;

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
```

(continues on next page)

(continued from previous page)

```

    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) NOT NULL DEFAULT ''
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";

DELIMITER :

CREATE TRIGGER "pdns_trig_records_insert"
AFTER INSERT ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_update"
AFTER UPDATE ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_delete"
AFTER DELETE ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = OLD."domain_id";
END;

DELIMITER ;

```

PostgreSQL

```

CREATE TABLE "domains" (
    "id" SERIAL NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "master" VARCHAR(40) NOT NULL DEFAULT '',
    "account" VARCHAR(40) NOT NULL DEFAULT '',
    "last_check" INTEGER DEFAULT NULL,
    "notified_serial" INTEGER DEFAULT NULL,
    "auto_serial" INTEGER NOT NULL DEFAULT 0,
    "status" CHAR(1) NOT NULL DEFAULT 'A',
    CONSTRAINT "pdns_pk_domains_id"
        PRIMARY KEY ("id"),
    CONSTRAINT "pdns_unq_domains_name"
        UNIQUE ("name")
);

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
    "id" SERIAL NOT NULL,
    "domain_id" INTEGER NOT NULL,

```

(continues on next page)

(continued from previous page)

```

    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "ttl" INTEGER DEFAULT NULL,
    "prio" INTEGER DEFAULT NULL,
    "content" VARCHAR(255) NOT NULL,
CONSTRAINT "pdns_pk_records_id"
    PRIMARY KEY ("id"),
CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
    REFERENCES "domains" ("id")
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) NOT NULL DEFAULT ''
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "domains_id_seq" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";
GRANT ALL ON "records_id_seq" TO "powerdns";

CREATE RULE "pdns_rule_records_insert"
AS ON INSERT TO "records" DO
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1 WHERE "id" = NEW.
↪ "domain_id";

CREATE RULE "pdns_rule_records_update"
AS ON UPDATE TO "records" DO
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1 WHERE "id" = NEW.
↪ "domain_id";

CREATE RULE "pdns_rule_records_delete"
AS ON DELETE TO "records" DO
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1 WHERE "id" = OLD.
↪ "domain_id";

```

SQLite and SQLite3

Supported without changes since OpenDBX 1.0.0 but requires to set *opendbx-host-read* to the path of the SQLite file (including the trailing slash or backslash, depending on your operating system) and *opendbx-database* to the name of the file.

```

opendbx-host-read = /path/to/file/
opendbx-host-write = /path/to/file/
opendbx-database = powerdns.sqlite

```

SQLite Schema

```

CREATE TABLE "domains" (
  "id" INTEGER NOT NULL PRIMARY KEY,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "master" VARCHAR(40) NOT NULL DEFAULT '',
  "account" VARCHAR(40) NOT NULL DEFAULT '',
  "last_check" INTEGER DEFAULT NULL,
  "notified_serial" INTEGER DEFAULT NULL,
  "auto_serial" INTEGER NOT NULL DEFAULT 0,
  "status" CHAR(1) NOT NULL DEFAULT 'A',
  CONSTRAINT "pdns_unq_domains_name"
    UNIQUE ("name")
);

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
  "id" INTEGER NOT NULL PRIMARY KEY,
  "domain_id" INTEGER NOT NULL,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "ttl" INTEGER DEFAULT NULL,
  "prio" INTEGER DEFAULT NULL,
  "content" VARCHAR(255) NOT NULL,
  CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
      REFERENCES "domains" ("id")
      ON UPDATE CASCADE
      ON DELETE CASCADE
);

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
  "ip" VARCHAR(40) NOT NULL,
  "nameserver" VARCHAR(255) NOT NULL,
  "account" VARCHAR(40) NOT NULL DEFAULT ''
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

CREATE TRIGGER "pdns_trig_records_insert"
AFTER INSERT ON "records"
FOR EACH ROW BEGIN
  UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
  WHERE "id" = NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_update"
AFTER UPDATE ON "records"
FOR EACH ROW BEGIN
  UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
  WHERE "id" = NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_delete"
AFTER DELETE ON "records"
FOR EACH ROW BEGIN
  UPDATE "domains" SET "auto_serial" = "auto_serial" + 1

```

(continues on next page)

(continued from previous page)

```
WHERE "id" = OLD."domain_id";
END;
```

SQLite3 Schema

```
CREATE TABLE "domains" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "master" VARCHAR(40) NOT NULL DEFAULT '',
  "account" VARCHAR(40) NOT NULL DEFAULT '',
  "last_check" INTEGER DEFAULT NULL,
  "notified_serial" INTEGER DEFAULT NULL,
  "auto_serial" INTEGER NOT NULL DEFAULT 0,
  "status" CHAR(1) NOT NULL DEFAULT 'A',
  CONSTRAINT "pdns_unq_domains_name"
    UNIQUE ("name")
);

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "domain_id" INTEGER NOT NULL,
  "name" VARCHAR(255) NOT NULL,
  "type" VARCHAR(6) NOT NULL,
  "ttl" INTEGER DEFAULT NULL,
  "prio" INTEGER DEFAULT NULL,
  "content" VARCHAR(255) NOT NULL,
  CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
      REFERENCES "domains" ("id")
      ON UPDATE CASCADE
      ON DELETE CASCADE
);

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
  "ip" VARCHAR(40) NOT NULL,
  "nameserver" VARCHAR(255) NOT NULL,
  "account" VARCHAR(40) NOT NULL DEFAULT ''
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

CREATE TRIGGER "pdns_trig_records_insert"
AFTER INSERT ON "records"
FOR EACH ROW BEGIN
  UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
  WHERE "id" = NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_update"
AFTER UPDATE ON "records"
FOR EACH ROW BEGIN
  UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
  WHERE "id" = NEW."domain_id";
```

(continues on next page)

(continued from previous page)

```
END;

CREATE TRIGGER "pdns_trig_records_delete"
AFTER DELETE ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = OLD."domain_id";
END;
```

Firebird/Interbase

Requires *opendbx-database* set to the path of the database file and doesn't support the default statement for starting transactions. Please add the following lines to your *pdns.conf*:

```
opendbx-database = /var/lib/firebird2/data/powerdns.gdb
opendbx-sql-transactbegin = SET TRANSACTION
```

When creating the database please make sure that you call the *isql* tool with the parameter `-page 4096`. Otherwise, you will get an error (key size exceeds implementation restriction for index "pdns_unq_domains_name") when creating the tables.

```
CREATE TABLE "domains" (
    "id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "master" VARCHAR(40) DEFAULT '' NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL,
    "last_check" INTEGER,
    "notified_serial" INTEGER,
    "auto_serial" INTEGER DEFAULT 0 NOT NULL,
    "status" CHAR(1) DEFAULT 'A' NOT NULL,
    CONSTRAINT "pdns_pk_domains_id"
        PRIMARY KEY ("id"),
    CONSTRAINT "pdns_unq_domains_name"
        UNIQUE ("name")
);

CREATE GENERATOR "pdns_gen_domains_id";

SET TERM !!;
CREATE TRIGGER "pdns_trig_domains_id" FOR "domains"
ACTIVE BEFORE INSERT AS
BEGIN
    IF (NEW."id" IS NULL) THEN
        NEW."id" = GEN_ID("pdns_gen_domains_id",1);
END !!
SET TERM ;!!

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
    "id" INTEGER NOT NULL,
    "domain_id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "ttl" INTEGER DEFAULT NULL,
    "prio" INTEGER DEFAULT NULL,
    "content" VARCHAR(255) NOT NULL,
    CONSTRAINT "pdns_pk_records_id"
```

(continues on next page)

(continued from previous page)

```
PRIMARY KEY ("id"),
CONSTRAINT "pdns_fk_records_domainid"
FOREIGN KEY ("domain_id")
REFERENCES "domains" ("id")
ON UPDATE CASCADE
ON DELETE CASCADE
);

CREATE GENERATOR "pdns_gen_records_id";

SET TERM !!;
CREATE TRIGGER "pdns_trig_records_id" FOR "records"
ACTIVE BEFORE INSERT AS
BEGIN
    IF (NEW."id" IS NULL) THEN
        NEW."id" = GEN_ID("pdns_gen_records_id",1);
    END !!
SET TERM ;!!

CREATE INDEX "idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";

SET TERM !!;

CREATE TRIGGER "pdns_trig_records_insert" FOR "records"
ACTIVE AFTER INSERT AS
BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = NEW."domain_id";
END !!

CREATE TRIGGER "pdns_trig_records_update" FOR "records"
ACTIVE AFTER UPDATE AS
BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = NEW."domain_id";
END !!

CREATE TRIGGER "pdns_trig_records_delete" FOR "records"
ACTIVE AFTER DELETE AS
BEGIN
    UPDATE "domains" d SET d."auto_serial" = d."auto_serial" + 1
    WHERE d."id" = OLD."domain_id";
END !!

SET TERM ;!!
```

Microsoft SQL Server

Supported using the FreeTDS library. It uses a different scheme for host configuration (requires the name of the host section in the configuration file of the dblib client library) and doesn't support the default statement for starting transactions. Please add the following lines to your pdns.conf:

```
opendbx-host-read = MSSQL2k
opendbx-host-write = MSSQL2k
opendbx-sql-transactbegin = BEGIN TRANSACTION
```

```
SET quoted_identifier ON;

CREATE TABLE "domains" (
    "id" INTEGER NOT NULL IDENTITY,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "master" VARCHAR(40) DEFAULT '' NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL,
    "last_check" INTEGER NULL,
    "notified_serial" INTEGER NULL,
    "auto_serial" INTEGER NOT NULL DEFAULT 0,
    "status" CHAR(1) DEFAULT 'A' NOT NULL,
    CONSTRAINT "pdns_pk_domains_id"
        PRIMARY KEY ("id"),
    CONSTRAINT "pdns_unq_domains_name"
        UNIQUE ("name")
);

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
    "id" INTEGER NOT NULL IDENTITY,
    "domain_id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "ttl" INTEGER NULL,
    "prio" INTEGER NULL,
    "content" VARCHAR(255) NOT NULL,
    "change_date" INTEGER NULL,
    CONSTRAINT "pdns_pk_records_id"
        PRIMARY KEY ("id"),
    CONSTRAINT "pdns_fk_records_domainid"
        FOREIGN KEY ("domain_id")
        REFERENCES "domains" ("id")
);

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL
);

CREATE INDEX "pdns_idx_smip_smns" ON "supermasters" ("ip","nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";
```

(continues on next page)

(continued from previous page)

```

CREATE TRIGGER "pdns_trig_records_insert"
ON "records" FOR INSERT AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT i."domain_id" FROM "inserted" i GROUP BY i."domain_id"
    );

CREATE TRIGGER "pdns_trig_records_update"
ON "records" FOR UPDATE AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT i."domain_id" FROM "inserted" i GROUP BY i."domain_id"
    );

CREATE TRIGGER "pdns_trig_records_delete"
ON "records" FOR DELETE AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT d."domain_id" FROM "deleted" d GROUP BY d."domain_id"
    );

```

Sybase ASE

Supported using the native Sybase ctlib or the FreeTDS library. It uses a different scheme for host configuration (requires the name of the host section in the configuration file of the ctlib client library) and doesn't support the default statement for starting transactions. Please add the following lines to your pdns.conf:

```

opendbx-host-read = SYBASE
opendbx-host-write = SYBASE
opendbx-sql-transactbegin = BEGIN TRANSACTION

```

```

SET quoted_identifier ON;

CREATE TABLE "domains" (
    "id" INTEGER NOT NULL IDENTITY,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "master" VARCHAR(40) DEFAULT '' NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL,
    "last_check" INTEGER NULL,
    "notified_serial" INTEGER NULL,
    "auto_serial" INTEGER NOT NULL DEFAULT 0,
    "status" CHAR(1) DEFAULT 'A' NOT NULL,
    CONSTRAINT "pdns_pk_domains_id"
        PRIMARY KEY ("id"),
    CONSTRAINT "pdns_unq_domains_name"
        UNIQUE ("name")
);

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status", "type");

CREATE TABLE "records" (
    "id" INTEGER NOT NULL IDENTITY,
    "domain_id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "ttl" INTEGER NULL,
    "prio" INTEGER NULL,
    "content" VARCHAR(255) NOT NULL,

```

(continues on next page)

(continued from previous page)

```

    "change_date" INTEGER NULL,
CONSTRAINT "pdns_pk_records_id"
    PRIMARY KEY ("id"),
CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
        REFERENCES "domains" ("id")
);

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name", "type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) DEFAULT '' NOT NULL
);

CREATE INDEX "pdns_idx_smip_smns" ON "supermasters" ("ip", "nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";

CREATE TRIGGER "pdns_trig_records_insert"
ON "records" FOR INSERT AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT i."domain_id" FROM "inserted" i GROUP BY i."domain_id"
    );

CREATE TRIGGER "pdns_trig_records_update"
ON "records" FOR UPDATE AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT i."domain_id" FROM "inserted" i GROUP BY i."domain_id"
    );

CREATE TRIGGER "pdns_trig_records_delete"
ON "records" FOR DELETE AS
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = ANY (
        SELECT d."domain_id" FROM "deleted" d GROUP BY d."domain_id"
    );

```

Oracle

Uses a different syntax for transactions and requires the following additional line in your pdns.conf:

```
opendbx-sql-transactbegin = SET TRANSACTION NAME 'AXFR'
```

```

CREATE TABLE "domains" (
    "id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "master" VARCHAR(40) DEFAULT '',
    "account" VARCHAR(40) DEFAULT '',
    "last_check" INTEGER,
    "notified_serial" INTEGER,
    "auto_serial" INTEGER DEFAULT 0,

```

(continues on next page)

(continued from previous page)

```

    "status" CHAR(1) DEFAULT 'A',
CONSTRAINT "pdns_pk_domains_id"
    PRIMARY KEY ("id"),
CONSTRAINT "pdns_unq_domains_name"
    UNIQUE ("name")
);

CREATE SEQUENCE "pdns_seq_domains_id" START WITH 1 INCREMENT BY 1;

CREATE TRIGGER "pdns_trig_domains_id"
BEFORE INSERT ON "domains"
FOR EACH ROW
BEGIN
    SELECT "pdns_seq_domains_id".nextval INTO :NEW."id" FROM dual;
END;

CREATE INDEX "pdns_idx_domains_status_type" ON "domains" ("status","type");

CREATE TABLE "records" (
    "id" INTEGER NOT NULL,
    "domain_id" INTEGER NOT NULL,
    "name" VARCHAR(255) NOT NULL,
    "type" VARCHAR(6) NOT NULL,
    "ttl" INTEGER NULL,
    "prio" INTEGER NULL,
    "content" VARCHAR(255) NOT NULL,
    "change_date" INTEGER NULL,
CONSTRAINT "pdns_pk_records_id"
    PRIMARY KEY ("id"),
CONSTRAINT "pdns_fk_records_domainid"
    FOREIGN KEY ("domain_id")
    REFERENCES "domains" ("id")
    ON DELETE CASCADE
);

CREATE SEQUENCE "pdns_seq_records_id" START WITH 1 INCREMENT BY 1;

CREATE TRIGGER "pdns_trig_records_id"
BEFORE INSERT ON "records"
FOR EACH ROW
BEGIN
    SELECT "pdns_seq_records_id".nextval INTO :NEW."id" FROM dual;
END;

CREATE INDEX "pdns_idx_records_name_type" ON "records" ("name","type");
CREATE INDEX "pdns_idx_records_type" ON "records" ("type");

CREATE TABLE "supermasters" (
    "ip" VARCHAR(40) NOT NULL,
    "nameserver" VARCHAR(255) NOT NULL,
    "account" VARCHAR(40) NOT NULL
);

CREATE INDEX "pdns_idx_smaster_ip_ns" ON "supermasters" ("ip","nameserver");

GRANT SELECT ON "supermasters" TO "powerdns";
GRANT ALL ON "domains" TO "powerdns";
GRANT ALL ON "records" TO "powerdns";

CREATE TRIGGER "pdns_trig_records_insert"
AFTER INSERT ON "records"

```

(continues on next page)

(continued from previous page)

```
FOR EACH ROW BEGIN
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = :NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_update"
AFTER UPDATE ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = :NEW."domain_id";
END;

CREATE TRIGGER "pdns_trig_records_delete"
AFTER DELETE ON "records"
FOR EACH ROW BEGIN
    UPDATE "domains" SET "auto_serial" = "auto_serial" + 1
    WHERE "id" = :OLD."domain_id";
END;
```

14.13 Oracle backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Autoserial: Yes
- DNSSEC: Yes
- Comments: No
- Module name: `oracle`
- Launch name: `oracle`

This is the Oracle Database backend 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.

14.13.1 Configuration Parameters

`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 `:nsname` 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.

14.13.2 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')"`.

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.

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 192.0.2.4:5300 notation for IPv4 and brackets for IPv6: [2001:db8::1234]:5300.

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.

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.

The Tables for Cryptographic Keys

We have two of them: ‘TSIGKeys’ for symmetric TSIG keys, and ‘ZoneDNSKeys’ for DNSSEC signing keys.

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.

ttd

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.

The SQL Statements

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
```

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 `pdnsutil` 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 `pdnsutil` 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
```

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 `pdnsutil` 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 `pdnsutil` 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 `pdnsutil` 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;
```

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;
```

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
```

Superslave Stuff

oracle-accept-supernotification-query

If a supernotification should be accepted from ‘:ip’, for the master nameserver ‘:ns’, return a label for this super-master. 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)
```

14.14 Pipe Backend

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

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

The PipeBackend is primarily meant for allowing rapid development of new backends without tight integration with PowerDNS. It allows end-users to write PowerDNS 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.

Note: The *Remote Backend* offers a superset of the functionality of the PipeBackend.

Note: Please do read the *Backend Writer’s guide* carefully. The PipeBackend, like all other backends, must not do any DNS thinking, but answer all questions (INCLUDING THE ANY QUESTION) faithfully. Specifically, the queries that the PipeBackend receives will not correspond to the queries that arrived over DNS. So, a query for

an AAAA record may turn into a backend query for an ANY record. There is nothing that can or should be done about this.

14.14.1 Configuration Parameters

`pipe-abi-version`

- Integer
- Default: 1

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 *pipe-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*). When set to 4 it sends zone name in AXFR request. See also *PipeBackend Protocol* below.

`pipe-command`

- String
- Mandatory

Command to launch as backend or the path to a unix domain socket file. The socket should already be open and listening before PowerDNS starts.

`pipe-timeout`

- Integer
- Default: 2000

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.

`pipe-regex`

- String (a regular expression)

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 'www.powerdns.com' would be presented to the regex as 'www.powerdns.com', a matching regex would be `^www\.powerdns\.com$`. **Note:** to match the root domain, use a dot, e.g. `^\. $`

14.14.2 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. Fields in lines must be separated by tab (`\t`) characters.

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.

Note: Fields are separated by a tab (`\t`) character, even though they are displayed with spaces in this document.

Q: Regular queries for data

The question format, for type Q questions.

pipe-abi-version = 1 [default]

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

pipe-abi-version = 2

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

pipe-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.

`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.

Note: Queries for wildcard names should be answered literally, without expansion. So, if a backend gets a question for `*.powerdns.com`, it should only answer with data if there is an actual `*.powerdns.com` name.

Note: In some (broken) network setups, the `remote-ip-address` and/or `local-ip-address`, when it is an IPv6 address, may be suffixed with a `%` and the name of the network interface (e.g. `%eth1`). Keep this in mind when checking the IP addresses.

AXFR: List an entire zone

AXFR-queries look like this:

AXFR	id	zone-name
------	----	-----------

The `id` is gathered from the answer to a SOA query. `zone-name` is given in ABI version 4.

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.

ABI version 1 and 2

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
------	-------	--------	-------	-----	----	---------

Again, all fields are tab-separated.

content is as specified in [Supported Record Types](#). 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.example.org IN  CNAME  -1  203.0.113.210
DATA  www.example.org IN  CNAME  3600  1  ws1.example.org
END
Q    ws1.example.org IN  CNAME  -1  203.0.113.210
END
Q    wsl.example.org IN  A      -1  203.0.113.210
DATA  wsl.example.org IN  A      3600  1  192.0.2.4
DATA  wsl.example.org IN  A      3600  1  192.0.2.5
DATA  wsl.example.org IN  A      3600  1  192.0.2.6
END
```

This would correspond to a remote webserver 203.0.113.210 wanting to resolve the IP address of www.example.org, and PowerDNS traversing the CNAMEs to find the IP addresses of ws1.example.org. Another dialogue might be:

```
Q    example.org      IN  SOA  -1  203.0.113.210
DATA  example.org      IN  SOA  86400  1  ahu.example.org ...
END
AXFR  1
DATA  example.org      IN  SOA  86400  1  ahu.example.org ...
DATA  example.org      IN  NS   86400  1  ns1.example.org
DATA  example.org      IN  NS   86400  1  ns2.example.org
DATA  ns1.example.org  IN  A     86400  1  203.0.113.210
DATA  ns2.example.org  IN  A     86400  1  63.123.33.135
.
.
END
```

This is a typical zone transfer.

ABI version 3 and higher

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. 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 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!

For abi-versions 1 and 2, the two new fields fall back to default values. The default value for `scopebits` is 0. The default for `auth` is 1 (meaning authoritative).

Direct backend commands

With abi-version 5 you can use *backend-cmd* for executing commands on your backend. PowerDNS will use the following query/answer format:

```
CMD      Whatever you wrote
Answer goes here
And can be multiple lines
until we see
END
```

14.14.3 Sample backends

ABI version 1

```
#!/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;
    }
}
```

(continues on next page)

(continued from previous page)

```

# note! the qname is what PowerDNS asks the backend. It need not be what
↳the internet asked PowerDNS!
my ($type,$qname,$qclass,$qtype,$id,$ip)=split(/\t/);

if(($qtype eq "SOA" || $qtype eq "ANY") && $qname eq "example.com") {
    print STDERR "$$ Sent SOA records\n";
    print "DATA      $qname
↳$qclass      SOA      3600      -1      ns1.example.com ahu.example.com
↳2008080300 1800 3600 604800 3600\n";
}
if(($qtype eq "NS" || $qtype eq "ANY") && $qname eq "example.com") {
    print STDERR "$$ Sent NS records\n";
    print "DATA      $qname
↳$qclass      NS      3600      -1      ns1.example.com\n";
    print "DATA      $qname
↳$qclass      NS      3600      -1      ns2.example.com\n";
}
if(($qtype eq "TXT" || $qtype eq "ANY") && $qname eq "example.com") {
    print STDERR "$$ Sent NS records\n";
    print "DATA      $qname
↳$qclass      TXT      3600      -1      \"hallo allemaal!\"\n";
}
if(($qtype eq "A" || $qtype eq "ANY") && $qname eq "webserver.example.com
↳") {
    print STDERR "$$ 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";
}
elseif(($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";
}

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

```

ABI version 3

```

#!/usr/bin/perl -w
# sample PowerDNS Coprocess backend with edns-client-subnet support
#

use strict;

$|=1;                                # no buffering

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

```

(continues on next page)

(continued from previous page)

```

unless($line eq "HELO\t3" ) {
    print "FAIL\n";
    print STDERR "Received unexpected '$line', wrong ABI version?\n";
    <>;
    exit;
}

print "OK          Sample backend firing up\n";          # print our banner

while(<>)
{
    print STDERR "$$ Received: $_";
    chomp();
    my @arr=split(/\t/);
    if(@arr < 8) {
        print "LOG          PowerDNS sent unparseable line\n";
        print "FAIL\n";
        next;
    }

    my ($type,$qname,$qclass,$qtype,$id,$ip,$localip,$ednsip)=split(/\t/);
    my $bits=21;
    my $auth = 1;

    if(($qtype eq "SOA" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent SOA records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          SOA          3600          -1          ahu.example.com ns1.example.com
↪2008080300 1800 3600 604800 3600\n";
    }
    if(($qtype eq "NS" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent NS records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          NS          3600          -1          ns1.example.com\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          NS          3600          -1          ns2.example.com\n";
    }
    if(($qtype eq "TXT" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent TXT records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          TXT          3600          -1          \"hallo allemaal!\"\n";
    }
    if(($qtype eq "A" || $qtype eq "ANY") && $qname eq "webserver.example.com
↪") {
        print STDERR "$$ Sent A records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          A          3600          -1          1.2.3.4\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          A          3600          -1          1.2.3.5\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          A          3600          -1          1.2.3.6\n";
    }
    if(($qtype eq "CNAME" || $qtype eq "ANY") && $qname eq "www.example.com") {
        print STDERR "$$ Sent CNAME records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          CNAME          3600          -1          webserver.example.com\n";
    }
    if(($qtype eq "MX" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent MX records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          MX          3600          -1          25          smtp.powerdns.com\n";
    }
}

```

(continues on next page)

(continued from previous page)

```

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

```

ABI version 5

```

#!/usr/bin/perl -w
# sample PowerDNS Coprocess backend with edns-client-subnet support
#

use strict;

$|=1;                                # no buffering

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

unless($line eq "HELO\t5" ) {
    print "FAIL\n";
    print STDERR "Received unexpected '$line', wrong ABI version?\n";
    <>;
    exit;
}

print "OK          Sample backend firing up\n";          # print our banner

while(<>)
{
    print STDERR "$$ Received: $_";
    chomp();
    my @arr=split(/\t/);

    if ($arr[0] eq "CMD") {
        print $arr[1], "\n";
        print "END\n";
        next;
    }

    if(@arr < 8) {
        print LOG          PowerDNS sent unparseable line\n";
        print "FAIL\n";
        next;
    }

    my ($type,$qname,$qclass,$qtype,$id,$ip,$localip,$ednsip)=split(/\t/);
    my $bits=21;
    my $auth = 1;

    if(($qtype eq "SOA" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent SOA records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          SOA          3600          -1          ahu.example.com ns1.example.com
↪2008080300 1800 3600 604800 3600\n";
    }

    if(($qtype eq "NS" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent NS records\n";
        print "DATA          $bits          $auth          $qname
↪$qclass          NS          3600          -1          ns1.example.com\n";
    }
}

```

(continues on next page)

(continued from previous page)

```

        print "DATA"      $bits      $auth      $qname
↪$qclass      NS          3600        -1          ns2.example.com\n";
    }
    if(($qtype eq "TXT" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent TXT records\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      TXT          3600        -1          \hallo allemaal!\n";
    }
    if(($qtype eq "A" || $qtype eq "ANY") && $qname eq "webserver.example.com
↪") {
        print STDERR "$$ Sent A records\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      A            3600        -1          1.2.3.4\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      A            3600        -1          1.2.3.5\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      A            3600        -1          1.2.3.6\n";
    }
    if(($qtype eq "CNAME" || $qtype eq "ANY") && $qname eq "www.example.com") {
        print STDERR "$$ Sent CNAME records\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      CNAME        3600        -1          webserver.example.com\n";
    }
    if(($qtype eq "MX" || $qtype eq "ANY") && $qname eq "example.com") {
        print STDERR "$$ Sent MX records\n";
        print "DATA"      $bits      $auth      $qname
↪$qclass      MX           3600        -1          25          smtp.powerdns.com\n";
    }

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

```

14.15 Random Backend

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- Autoserial: No
- Case: Depends
- DNSSEC: Yes, no key storage
- Disabled data: No
- Comments: No
- Module name: built in
- Launch: random

This is a very silly backend which is discussed in the *Backends writer's guide*, 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`.

14.15.1 Configuration Parameters

`random-hostname`

- String

Hostname for which to supply a random IP address.

14.16 Remote Backend

- Native: Yes
- Master: Yes*
- Slave: Yes*
- Superslave: Yes*
- Autoserial: Yes*
- DNSSEC: Yes*
- Multiple instances: Yes

* If provided by the responder (your script).

This backend provides Unix socket, Pipe, HTTP and ZeroMQ remoting for powerdns. You should think this as normal RPC thin client, which converts native C++ calls into JSON/RPC and passes them to you via connector.

14.16.1 Important notices

Please do not use remotebackend shipped before version 3.3. This version has severe bug that can crash the entire process.

There is a breaking change on v4.0 and later. Before version 4.0, the DNS names passed in queries were without trailing dot, after version 4.0 the DNS names are sent with trailing dot. F.ex. `example.org` is now sent as `example.org.`

In some (broken) network setups, the IP addresses provided in the request (when this is an IPv6 address) may be suffixed with a `%` and the name of the network interface (e.g. `%eth1`). Keep this in mind when checking the IP addresses.

14.16.2 Compiling

To compile this backend, you need to configure `--with-modules="remote"`.

For versions prior to 3.4.0, if you want to use http connector, you need libcurl and use `--enable-remotebackend-http`.

If you want to use ZeroMQ connector, you need libzmq-dev or libzmq3-dev and use `--enable-remotebackend-zeromq`.

14.16.3 Usage

The only configuration options for backend are `remote-connection-string` and `remote-dnssec`.

```
remote-connection-string=<type>:<param>=<value>,<param>=<value>...
```

You can pass as many parameters as you want. For unix and pipe connectors, these are passed along to the remote end as initialization. See [API](#). Initialize is not called for http connector.

Unix connector

parameters: path, timeout (default 2000ms)

```
remote-connection-string=unix:path=/path/to/socket
```

Pipe connector

parameters: command, timeout (default 2000ms)

```
remote-connection-string=pipe:command=/path/to/executable,timeout=2000
```

HTTP connector

parameters: url, url-suffix, post, post_json, timeout (default 2000ms)

```
remote-connection-string=http:url=http://localhost:63636/dns,url-suffix=.php
```

HTTP connector tries to do RESTful requests to your server. See examples. You can also use `post` to change behaviour so that it will send POST request to `url/method + url_suffix` with `parameters=json-formatted-parameters`. If you use `post` and `post_json`, it will POST url with text/javascript containing JSON formatted RPC request, just like for pipe and unix. You can use '1', 'yes', 'on' or 'true' to turn these features on.

URL should not end with /, and url-suffix is optional, but if you define it, it's up to you to write the ".php" or ".json". Lack of dot causes lack of dot in URL. Timeout is divided by 1000 because libcurl only supports seconds, but this is given in milliseconds for consistency with other connectors.

HTTPS is not supported, [stunnel](#) is the suggested workaround. HTTP Authentication is not supported.

ZeroMQ connector

parameters: endpoint, timeout (default 2000ms)

```
remote-connection-string=zeromq:endpoint=ipc:///tmp/tmp.sock
```

OMQ connector implements a REQ/REP RPC model. Please see <http://zeromq.org/> for more information.

14.16.4 API

Queries

Unix, Pipe and ZeroMQ connectors send JSON formatted strings to the remote end. Each JSON query has two sections, 'method' and 'parameters'.

HTTP connector calls methods based on URL and has parameters in the query string. Most calls are GET; see the methods listing for details. You can change this with `post` and `post_json` attributes.

Replies

You **must** always reply with JSON hash with at least one key, 'result'. This must be boolean false if the query failed. Otherwise it must conform to the expected result. For HTTP connector, to signal bare success, you can just reply with HTTP 200 OK, and omit any output. This will result in same outcome as sending {"result":true}.

You can optionally add an array of strings to the 'log' array; each line in this array will be logged in PowerDNS at loglevel info (6).

Methods

initialize

Called to initialize the backend. This is not called for HTTP connector. You should do your initializations here.

- Mandatory: Yes (except HTTP connector)
- Parameters: all parameters in connection string
- Reply: true on success / false on failure

Example JSON/RPC

Query:

```
{ "method": "initialize", "parameters": { "command": "/path/to/something", "timeout":  
↪ "2000", "something": "else" } }
```

Response:

```
{ "result": true }
```

lookup

This method is used to do the basic query. You can omit auth, but if you are using DNSSEC this can lead into trouble.

- Mandatory: Yes
- Parameters: qtype, qname, zone_id
- Optional parameters: remote, local, real-remote
- Reply: array of qtype, qname, content, ttl, domain_id, scopeMask, auth
- Optional values: domain_id, scopeMask and auth
- Note: priority field is required before 4.0, after 4.0 priority is added to content. This applies to any resource record which uses priority, for example SRV or MX.

Example JSON/RPC

Query:

```
{ "method": "lookup", "parameters": { "qtype": "ANY", "qname": "www.example.com.",  
↪ "remote": "192.0.2.24", "local": "192.0.2.1", "real-remote": "192.0.2.24", "zone-id  
↪ ": -1 } }
```

Response:

```
{ "result": [{ "qtype": "A", "qname": "www.example.com", "content": "203.0.113.2", "ttl": 60 } ] }
```

Example HTTP/RPC

Query:

```
GET /dnsapi/lookup/www.example.com./ANY HTTP/1.1
X-RemoteBackend-remote: 192.0.2.24
X-RemoteBackend-local: 192.0.2.1
X-RemoteBackend-real-remote: 192.0.2.24
X-RemoteBackend-zone-id: -1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{ "result": [{ "qtype": "A", "qname": "www.example.com", "content": "203.0.113.2", "ttl": 60 } ] }
```

list

Lists all records for the zonename. If you are running dnssec, you should take care of setting auth to appropriate value, otherwise things can go wrong.

- Mandatory: No (Gives AXFR support)
- Parameters: zonename, domain_id
- Optional parameters: domain_id
- Reply: array of qtype, qname, content, ttl, domain_id, scopeMask, auth
- Optional values: domain_id, scopeMask and auth

Example JSON/RPC

Query:

```
{ "method": "list", "parameters": { "zonename": "example.com.", "domain_id": -1 } }
```

Response (split into lines for ease of reading)

```
{ "result": [
  { "qtype": "SOA", "qname": "example.com", "content": "dns1.icann.org. hostmaster.
  ↪ icann.org. 2012081600 7200 3600 1209600 3600", "ttl": 3600 },
  { "qtype": "NS", "qname": "example.com", "content": "ns1.example.com", "ttl": 60 },
  { "qtype": "MX", "qname": "example.com", "content": "10 mx1.example.com.", "ttl": 60 }
  ↪,
  { "qtype": "A", "qname": "www.example.com", "content": "203.0.113.2", "ttl": 60 },
  { "qtype": "A", "qname": "ns1.example.com", "content": "192.0.2.2", "ttl": 60 },
  { "qtype": "A", "qname": "mx1.example.com", "content": "192.0.2.3", "ttl": 60 }
] }
```

Example HTTP/RPC

Query:

```
GET /dnsapi/list/-1/example.com HTTP/1.1
X-RemoteBackend-domain-id: -1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":[{"qtype":"SOA", "qname":"example.com", "content":"dns1.icann.org.↵
↵hostmaster.icann.org. 2012081600 7200 3600 1209600 3600", "ttl": 3600}, {"qtype":
↵"NS", "qname":"example.com", "content":"ns1.example.com", "ttl": 60}, {"qtype":"MX
↵", "qname":"example.com", "content":"10 mx1.example.com.", "ttl": 60}, {"qtype":"A
↵", "qname":"www.example.com", "content":"203.0.113.2", "ttl": 60}, {"qtype":"A",
↵"qname":"ns1.example.com", "content":"192.0.2.2", "ttl": 60}, {"qtype":"A", "qname
↵":"mx1.example.com", "content":"192.0.2.3", "ttl": 60}]}
```

getBeforeAndAfterNamesAbsolute

Asks the names before and after qname. qname is given without dots or domain part. The query will be hashed when using NSEC3. Care must be taken to handle wrap-around when qname is first or last in the ordered list. Do not return nil for either one.

- Mandatory: for NSEC/NSEC3 non-narrow
- Parameters: id, qname
- Reply: before, after

Example JSON/RPC

Query:

```
{"method":"getbeforeandafternamesabsolute", "params":{"id":0,"qname":"www.example.
↵com"}}
```

Response:

```
{"result":{"before":"ns1","after":""}}
```

Example HTTP/RPC

Query:

```
/dnsapi/getbeforeandafternamesabsolute/0/www.example.com
```

Response:

```
{"result":{"before":"ns1","after":""}}
```

getAllDomainMetadata

Returns the value(s) for variable kind for zone name. You **must** always return something, if there are no values, you shall return empty set or false.

- Mandatory: No
- Parameters: name
- Reply: hash of key to array of strings

Example JSON/RPC

Query:

```
{ "method": "getalldomainmetadata", "parameters": { "name": "example.com" } }
```

Response:

```
{ "result": { "PRESIGNED": [ "0" ] } }
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getalldomainmetadata/example.com HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{ "result": { "PRESIGNED": [ "0" ] } }
```

getDomainMetadata

Returns the value(s) for variable kind for zone name. Most commonly it's one of NSEC3PARAM, PRESIGNED, SOA-EDIT. Can be others, too. You **must** always return something, if there are no values, you shall return empty array or false.

- Mandatory: No
- Parameters: name, kind
- Reply: array of strings

Example JSON/RPC

Query:

```
{ "method": "getdomainmetadata", "parameters": { "name": "example.com.", "kind":  
  ↪ "PRESIGNED" } }
```

Response:

```
{ "result": [ "0" ] }
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getdomainmetadata/example.com./PRESIGNED HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":["0"]}
```

setDomainMetadata

Replaces the value(s) on domain name for variable kind to string(s) on array value. The old value is discarded. Value can be an empty array, which can be interpreted as deletion request.

- Mandatory: No
- Parameters: name, kind, value
- Reply: true on success, false on failure

Example JSON/RPC

Query:

```
{"method": "setdomainmetadata", "parameters": {"name": "example.com", "kind": "PRESIGNED", "value": ["YES"]}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/setdomainmetadata/example.com./PRESIGNED HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 12

value[]=YES&
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": true}
```

getDomainKeys

Retrieves any keys of kind. The id, flags are unsigned integers, and active is boolean. Content must be valid key record in format that PowerDNS understands. You are encouraged to implement *the section called “addDomainKey”*, as you can use *pdnsutil* to provision keys.

- Mandatory: for DNSSEC
- Parameters: name, kind

- Reply: array of id, flags, active, content

Example JSON/RPC

Query:

```
{ "method": "getdomainkeys", "parameters": { "name": "example.com." } }
```

Response:

```
{ "result": [ { "id": 1, "flags": 256, "active": true, "content": "Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:
↳ r+vmQ1l38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt /
↳ g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy /
↳ u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+ /
↳ 6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jj04jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↳ E=
Prime1: 6a165cIC0nNsG1TW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd /
↳ Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6 /
↳ 9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX /
↳ Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTIQ0yF /
↳ 3Baj5NAi9w84l0u07KgKQZX4g0N8qUyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEULZh96m8WiaE1 /
↳ fIt4Zl2PC3ff7YIBoFLln22w==" ] ] }
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getdomainkeys/example.com/0 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{ "result": [ { "id": 1, "flags": 256, "active": true, "content": "Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:
↳ r+vmQ1l38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt /
↳ g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy /
↳ u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+ /
↳ 6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jj04jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↳ E=
Prime1: 6a165cIC0nNsG1TW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd /
↳ Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6 /
↳ 9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX /
↳ Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
```

(continues on next page)

(continued from previous page)

```
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪3Baj5NAi9w84l0u07KgKQZX4g0N8qUyyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEU1Zh96m8WiaE1/
↪fIt4Zl2PC3ff7YIBoFLln22w=="}}}

```

addDomainKey

Adds key into local storage. See *getDomainKeys* for more information.

- Mandatory: No
- Parameters: name, key=<flags, active, content>, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{ "method": "adddomainkey", "parameters": { "key": { "id": 1, "flags": 256, "active": true,
↪ "content": "Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus: ↪
↪ r+vmQ1l38ndQqNSCx9eqRBUBsOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪ g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCF8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪ u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↪ 6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRW5jj04jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↪ E=
Prime1: 6a165cIC0nNsG1TW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRumq+4H9ASd/
↪ Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↪ 9IKOYz8t9ZNLy74iCeQ85koqvTctkFmB9jpoUHTU9BhecaFY2euP9CuHV7z3PLtCo08s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↪ Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪ 3Baj5NAi9w84l0u07KgKQZX4g0N8qUyyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEU1Zh96m8WiaE1/
↪ fIt4Zl2PC3ff7YIBoFLln22w=="}}}

```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
PUT /dnsapi/adddomainkey/example.com HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 965

flags=256&active=1&content=Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus: ↪
↪ r+vmQ1l38ndQqNSCx9eqRBUBsOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪ g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCF8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪ u7UN1oHKSKRWuZJadUwcjbp8=

```

(continues on next page)

(continued from previous page)

```

PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↳ 6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jj04jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↳ E=
Prime1: 6a165cIC0nNsG1TW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd/
↳ Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↳ 9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↳ Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQITQ0yF/
↳ 3Baj5NAi9w84l0u07KgKQZX4g0N8qUyyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEU1Zh96m8WiaE1/
↳ fIt4Zl2PC3fF7YIBoFLln22w==

```

Response:

```

HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}

```

removeDomainKey

Removes key id from domain name.

- Mandatory: No
- Parameters: name, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{ "method": "removedomainkey", "parameters": { "name": "example.com", "id": 1 } }
```

Response:

```
{ "result": true }
```

Example HTTP/RPC

Query:

```
DELETE /dnsapi/removedomainkey/example.com/1 HTTP/1.1
```

Response:

```

HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}

```

`activateDomainKey`

Activates key id for domain name.

- Mandatory: No
- Parameters: name, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "activatedomainkey", "parameters": {"name": "example.com", "id": 1}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/activatedomainkey/example.com/1 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; utf-8

{"result": true}
```

`deactivateDomainKey`

Deactivates key id for domain name.

- Mandatory: No
- Parameters: name, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "deactivatedomainkey", "parameters": {"name": "example.com", "id": 1}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/deactivatedomainkey/example.com/1 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; utf-8

{"result": true}
```

getTSIGKey

Retrieves the key needed to sign AXFR.

- Mandatory: No
- Parameters: name
- Reply: algorithm, content

Example JSON/RPC

Query:

```
{"method": "gettsigkey", "parameters": {"name": "example.com."}}
```

Response:

```
{"result": {"algorithm": "hmac-md5", "content": "kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/↵3oRSP7ys="}}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/gettsigkey/example.com. HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": {"algorithm": "hmac-md5", "content": "kp4/24gyYsEzbuTVJRUMoqGFmN3LYgVDzJ/↵3oRSP7ys="}}
```

getDomainInfo

Retrieves information about given domain from the backend. If your return value has no zone attribute, the backend will signal error. Everything else will default to something. Default values: serial:0, kind:NATIVE, id:-1, notified_serial:-1, last_check:0, masters: []. Masters, if present, must be array of strings.

- Mandatory: No
- Parameters: name

- Reply: zone
- Optional values: serial, kind, id, notified_serial, last_check, masters

Example JSON/RPC

Query:

```
{"method": "getdomaininfo", "parameters": {"name": "example.com"}}
```

Response:

```
{"result": {"id": 1, "zone": "example.com", "kind": "NATIVE", "serial": 2002010100}}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getdomaininfo/example.com HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
content-Type: text/javascript; charset=utf-8

{"result": {"id": 1, "zone": "example.com", "kind": "NATIVE", "serial": 2002010100}}
```

setNotified

Updates last notified serial for the domain id. Any errors are ignored.

- Mandatory: No
- Parameters: id, serial
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "setnotified", "parameters": {"id": 1, "serial": 2002010100}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/setnotified/1 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 17

serial=2002010100
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

isMaster

Determines whether given IP is master for given domain name.

- Mandatory: No
- Parameters: name,ip
- Reply: true for success, false for failure.

Example JSON/RPC

Query:

```
{"method":"isMaster","parameters":{"name":"example.com","ip":"198.51.100.0.1"}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/isMaster/example.com/198.51.100.0.1 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

superMasterBackend

Creates new domain with given record(s) as master servers. IP address is the address where notify is received from. nsset is array of NS resource records.

- Mandatory: No
- Parameters: ip,domain,nsset,account
- Reply: true for success, false for failure. can also return account=>name of account< and nameserver.

Example JSON/RPC

Query:

```
{"method":"superMasterBackend","parameters":{"ip":"198.51.100.0.1","domain":
↪ "example.com","nsset":[{"qtype":"NS","qname":"example.com","qclass":1,"content":
↪ "ns1.example.com","ttl":300,"auth":true}, {"qtype":"NS","qname":"example.com",
↪ "qclass":1,"content":"ns2.example.com","ttl":300,"auth":true}]}}
```

(continues on next page)

(continued from previous page)

Response:

```
{"result":true}
```

Alternative response:

```
{"result":{"account":"my account","nameserver":"ns2.example.com"}}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/supermasterbackend/198.51.100.0.1/example.com HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 317

nsset[1][qtype]=NS&nsset[1][qname]=example.com&nsset[1][qclass]=1&
↪nsset[1][content]=ns1.example.com&nsset[1][ttl]=300&nsset[1][auth]=true&
↪nsset[2][qtype]=NS&nsset[2][qname]=example.com&nsset[2][qclass]=1&
↪nsset[2][content]=ns2.example.com&nsset[2][ttl]=300&nsset[2][auth]=true
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

Alternative response

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":{"account":"my account"}}
```

createSlaveDomain

Creates new domain. This method is called when NOTIFY is received and you are superslaving.

Mandatory: No Parameters: ip, domain Optional parameters: nameserver, account Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"createSlaveDomain","parameters":{"ip":"198.51.100.0.1","domain":"pirate.
↪example.net"}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/createslavedomain/198.51.100.0.1/pirate.example.net HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

replaceRRSet

This method replaces a given resource record with new set. The new qtype can be different from the old.

- Mandatory: No
- Parameters: domain_id, qname, qtype, rrset
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"replaceRRSet","parameters":{"domain_id":2,"qname":"replace.example.com",
↪ "qtype":"A","trxid":1370416133,"rrset":[{"qtype":"A","qname":"replace.example.com
↪ ","qclass":1,"content":"1.1.1.1","ttl":300,"auth":true}]}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/replacerrset/2/replace.example.com/A HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 135

trxid=1370416133&rrset[qtype]=A&rrset[qname]=replace.example.com&rrset[qclass]=1&
↪ rrset[content]=1.1.1.1&rrset[auth]=1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

feedRecord

Asks to feed new record into system. If startTransaction was called, trxId identifies a transaction. It is not always called by PowerDNS.

- Mandatory: No
- Parameters: rr, trxid
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "feedRecord", "parameters": {"rr": {"qtype": "A", "qname": "replace.example.com", "qclass": 1, "content": "127.0.0.1", "ttl": 300, "auth": true}, "trxid": 1370416133}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/feedrecord/1370416133 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 117

rr[qtype]=A&rr[qname]=replace.example.com&rr[qclass]=1&rr[content]=127.0.0.1&rr[ttl]=300&rr[auth]=true
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": true}
```

feedEnts

This method is used by pdnsutil rectify-zone to populate missing non-terminals. This is used when you have, say, record like _sip._udp.example.com, but no _udp.example.com. PowerDNS requires that there exists a non-terminal in between, and this instructs you to add one. If startTransaction is called, trxid identifies a transaction.

- Mandatory: No
- Parameters: nonterm, trxid
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "feedEnts", "parameters": {"domain_id": 2, "trxid": 1370416133, "nonterm": ["_sip._udp", "_udp"]}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/feedents/2 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

trxid=1370416133&nonterm[]=_udp&nonterm[]=_sip.udp
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

feedEnts3

Same as *feedEnts*, but provides NSEC3 hashing parameters. Note that salt is BYTE value, and can be non-readable text.

- Mandatory: No
- Parameters: trxid, domain_id, domain, times, salt, narrow, nonterm
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"feedEnts3","parameters":{"domain_id":2,"domain":"example.com","times":1,
↪ "salt":"9642","narrow":false,"trxid":1370416356,"nonterm":["_sip._udp","_udp"]}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/2/example.com HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 78

trxid=1370416356&times=1&salt=9642&narrow=0&nonterm[]=_sip._udp&nonterm[]=_udp
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

`startTransaction`

Starts a new transaction. Transaction ID is chosen for you. Used to identify f.ex. AXFR transfer.

- Mandatory: No
- Parameters: domain_id, domain, trxid
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "startTransaction", "parameters": {"trxid": 1234, "domain_id": 1, "domain": "example.com"}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/starttransaction/1/example.com HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

trxid=1234
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

`commitTransaction`

Signals successful transfer and asks to commit data into permanent storage.

- Mandatory: No
- Parameters: trxid
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "commitTransaction", "parameters": {"trxid": 1234}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/committransaction/1234 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": true}
```

abortTransaction

Signals failed transaction, and that you should rollback any changes.

- Mandatory: No
- Parameters: trxid
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method": "abortTransaction", "parameters": {"trxid": 1234}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/aborttransaction/1234 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":true}
```

calculateSOASerial

Asks you to calculate a new serial based on the given data and update the serial.

- Mandatory: No
- Parameters: domain,sd
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{ "method": "calculateSOASerial", "parameters": { "domain": "unit.test", "sd": { "qname":
↪ "unit.test", "nameserver": "ns.unit.test", "hostmaster": "hostmaster.unit.test", "ttl
↪ ": 300, "serial": 1, "refresh": 2, "retry": 3, "expire": 4, "default_ttl": 5, "domain_id": -1,
↪ "scopeMask": 0 } } }
```

Response:

```
{ "result": 2013060501 }
```

Example HTTP/RPC

Query:

```
POST /dnsapi/calculatesoaserial/unit.test HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 198

sd[qname]=unit.test&sd[nameserver]=ns.unit.test&sd[hostmaster]=hostmaster.unit.
↪ test&sd[ttl]=300&sd[serial]=1&sd[refresh]=2&sd[retry]=3&sd[expire]=4&sd[default_
↪ ttl]=5&sd[domain_id]=-1&sd[scopemask]=0
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result":2013060501}
```

directBackendCmd

Can be used to send arbitrary commands to your backend using *pdnsutil* and *DNSSEC*.

- Mandatory: no
- Parameters: query
- Reply: anything but boolean false for success, false for failure

Example JSON/RPC

Query:

```
{"method": "directBackendCmd", "parameters": {"query": "PING"}}
```

Response:

```
{"result": "PONG"}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/directBackendCmd HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

query=PING
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": "PONG"}
```

getAllDomains

Get DomainInfo records for all domains in your backend.

- Mandatory: no
- Parameters: include_disabled
- Reply: array of DomainInfo

Example JSON/RPC

Query:

```
{"method": "getAllDomains", "parameters": {"include_disabled": true}}
```

Response:

```
{"result": [{"id": 1, "zone": "unit.test.", "masters": ["10.0.0.1"], "notified_serial": 2,
↪ "serial": 2, "last_check": 1464693331, "kind": "native"}]}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getAllDomains?includeDisabled=true HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8
Content-Length: 135
{"result":[{"id":1,"zone":"unit.test.","masters":["10.0.0.1"],"notified_serial":2,
↪ "serial":2,"last_check":1464693331,"kind":"native"}]}
```

searchRecords

Can be used to search records from the backend. This is used by web api.

- Mandatory: no
- Parameters: pattern, maxResults
- Reply: same as *lookup* or false to indicate failed search

Example JSON/RPC

Query:

```
{"method": "searchRecords", "parameters": {"pattern": "www.example*", "maxResults": 100}}
```

Response:

```
{"result": [{"qtype": "A", "qname": "www.example.com", "content": "203.0.113.2", "ttl": 60}]}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/searchRecords?q=www.example*&maxResults=100 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8

{"result": [{"qtype": "A", "qname": "www.example.com", "content": "203.0.113.2", "ttl": 60}]}
```

14.16.5 Examples

Scenario: SOA lookup via pipe, unix or zeromq connector

Query:

```
{
  "method": "lookup",
  "parameters": {
    "qname": "example.com",
    "qtype": "SOA",
    "zone_id": "-1"
  }
}
```

Reply:

```
{
  "result":
  [
    { "qtype": "SOA",
      "qname": "example.com",
      "content": "dns1.icann.org. hostmaster.icann.org. 2012080849 7200 3600_
↪1209600 3600",
      "ttl": 3600,
      "domain_id": -1
    }
  ]
}
```

Scenario: SOA lookup with HTTP connector

Query:

```
/dns/lookup/example.com/SOA
```

Reply:

```
{
  "result":
  [
    { "qtype": "SOA",
      "qname": "example.com",
      "content": "dns1.icann.org. hostmaster.icann.org. 2012080849 7200 3600_
↪1209600 3600",
      "ttl": 3600,
      "domain_id": -1
    }
  ]
}
```

14.17 TinyDNS Backend

- Native: Yes
- Master: Yes
- Slave: No
- Superslave: No
- Autoserail: No
- DNSSEC: No
- Multiple Instances: Yes
- Module name: tinydns
- Launch: tinydns

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.

14.17.1 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`

- String
- Default: data.cdb

Specifies the name of the data file to use.

`tinydns-tai-adjust`

- Integer
- Default: 11

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 11. The last update was on [june 30th 2012](#).

`tinydns-notify-on-startup`

- Boolean
- Default: no

Tell the TinyDNSBackend to notify all the slave nameservers on startup. This might cause broadcast storms.

`tinydns-ignore-bogus-records`

- Boolean
- Default: no

The `tinydns-data` program can create data.cdb files that have bad/corrupt RDATA. PowerDNS will crash when it tries to read that bad/corrupt data. This option (change to yes), allows you to ignore that bad RDATA to make PowerDNS operate when bad data is in your CDB file. Be aware that the records are then ignored, where tinydns would still send out the bogus data. The option is primarily useful in master mode, as that reads all the packets in the zone to find all the SOA records.

`tinydns-locations`

- Boolean
- Default: yes

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.

14.17.2 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 *Performance and Tuning*. Location support only exists for IPv4!

14.17.3 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.

14.17.4 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](#).

PowerDNS and TinyDNS handle wildcards differently. Looking up `foo.www.example.com` with the below records on TinyDNS will return `198.51.100.1`, PowerDNS will return `NXDOMAIN`. According to [RFC 4592](#) `*.example.com` should only match subdomains in under `example.com`, not `*.*.example.com`. This compatibility issue is noted on the [axfer-get](#) page for the `djbdns` suite.

<code>*.example.com</code>	<code>A 198.51.100.1</code>
<code>www.example.com</code>	<code>A 198.51.100.1</code>

Compiling the TinyDNS backend requires you to have `tinycdb` version 0.77.

BUILT-IN WEBSERVER AND HTTP API

The PowerDNS Authoritative Server features a built-in webserver that exposes a JSON/REST API. This API allows for controlling several functions, reading statistics and modifying zone content, metadata and DNSSEC key material.

15.1 Webserver

To launch the internal webserver, add a *webserver* to the configuration file. This will instruct PowerDNS to start a webserver on localhost at port 8081, without password protection. By default the webserver listens on localhost, meaning only local users (on the same host) will be able to access the webserver. Since the default ACL before 4.1.0 allows access from everywhere if *webserver-address* is set to a different value, we strongly advise the use of a password protection. The webserver lists a lot of potentially sensitive information about the PowerDNS 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, but allows control over the size of the queries and response rings that may be used to monitor activities. The following webserver 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, in addition to entering the configured API key on the index page.
- *webserver-port*: Port to bind the webserver to.
- *webserver-allow-from*: Netmasks that are allowed to connect to the webserver

15.2 Enabling the API

To enable the API, the webserver and the HTTP API need to be enabled. Add these lines to the `pdns.conf`:

```
api=yes
api-key=changeme
```

Changed in version 4.1.0: Setting *api* also implicitly enables the webserver.

And restart, the following examples should start working:

```
curl -v -H 'X-API-Key: changeme' http://127.0.0.1:8081/api/v1/servers/localhost | jq .
curl -v -H 'X-API-Key: changeme' http://127.0.0.1:8081/api/v1/servers/localhost/
zones | jq .
```

15.3 Working with the API

This chapter describes the PowerDNS Authoritative API. When creating an API wrapper (for instance when fronting multiple API's), it is recommended to stick to this API specification. The API is described in the [OpenAPI format](#), also known as “Swagger”, and this description is [available](#).

15.3.1 Authentication

The PowerDNS daemons accept a static API Key, configured with the *api-key* option, which has to be sent in the `X-API-Key` header.

15.3.2 Errors

Response code 4xx or 5xx, depending on the situation.

- Invalid JSON body from client: 400 Bad Request
- Input validation failed: 422 Unprocessable Entity
- JSON body from client not a hash: 400 Bad Request

Error responses have a JSON body of this format:

```
{
  "error": "short error message",
  "errors": [
    { },
  ]
}
```

Where `errors` is optional, and the contents are error-specific.

15.3.3 Data format

The API accepts and emits **JSON**. The `Accept:` header determines the output format. An unknown value or `*/*` will cause a 400 Bad Request.

All text is UTF-8 and HTTP headers will reflect this.

Data types:

- empty fields: `null` but present
- Regex: implementation defined
- Dates: ISO 8601

15.4 Endpoints and Objects in the API

The API has the basepath `/api/v1` and all URLs in this documentation are relative to this basepath.

The API exposes several endpoints and objects:

15.4.1 Servers

The server endpoint is the ‘basis’ for all other API operations. In the PowerDNS Authoritative Server, the `server_id` is always `localhost`. However, the API is written in a way that a proxy could be in front of many servers, each with their own `server_id`.

Endpoints

GET /servers

List all servers

Status Codes

- **200 OK** – An array of servers Returns: array of *Server* objects

GET /servers/{server_id}

List a server

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Status Codes

- **200 OK** – An server Returns: *Server* object

Objects

Server

Object Properties

- **type** (*string*) – Set to “Server”
- **id** (*string*) – The id of the server, “localhost”
- **daemon_type** (*string*) – “recursor” for the PowerDNS Recursor and “authoritative” for the Authoritative Server
- **version** (*string*) – The version of the server software
- **url** (*string*) – The API endpoint for this server
- **config_url** (*string*) – The API endpoint for this server’s configuration
- **zones_url** (*string*) – The API endpoint for this server’s zones

15.4.2 Zones

Manipulating zones is the primary use of the API.

Zone Endpoints

GET /servers/{server_id}/zones

List all Zones in a server

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Status Codes

- **200 OK** – An array of Zones Returns: array of *Zone* objects

POST /servers/{server_id}/zones

Creates a new domain, returns the Zone on creation.

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Query Parameters

- **rrsets** (*boolean*) – “true” (default) or “false”, whether to include the “rrsets” in the response Zone object.

Status Codes

- **201 Created** – A zone Returns: *Zone* object

GET /servers/{server_id}/zones/{zone_id}
zone managed by a server

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- **200 OK** – A Zone Returns: *Zone* object

DELETE /servers/{server_id}/zones/{zone_id}
Deletes this zone, all attached metadata and rrsets.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- **200 OK** – OK

PATCH /servers/{server_id}/zones/{zone_id}
Creates/modifies/deletes RRsets present in the payload and their comments. Returns 204 No Content on success.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –

Status Codes

- **204 No Content** – Returns 204 No Content on success.

PUT /servers/{server_id}/zones/{zone_id}
Modifies basic zone data (metadata).

Allowed fields in client body: all except id, url and name. Returns 204 No Content on success.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –

Status Codes

- **204 No Content** – Returns 204 No Content on success.

PUT /servers/{server_id}/zones/{zone_id}/axfr-retrieve
Send a DNS NOTIFY to all slaves.

Fails when zone kind is not Master or Slave, or master and slave are disabled in the configuration. Only works for Slave if renotify is on. Clients MUST NOT send a body.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- 200 OK – OK

PUT /servers/{server_id}/zones/{zone_id}/notify
Send a DNS NOTIFY to all slaves.

Fails when zone kind is not Master or Slave, or master and slave are disabled in the configuration. Only works for Slave if renotify is on. Clients MUST NOT send a body.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- 200 OK – OK

GET /servers/{server_id}/zones/{zone_id}/export
Returns the zone in AXFR format.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- 200 OK – OK Returns: string

GET /servers/{server_id}/zones/{zone_id}/check
Verify zone contents/configuration.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- 200 OK – OK Returns: string

PUT /servers/{server_id}/zones/{zone_id}/rectify
Rectify the zone data.

This does not take into account the API-RECTIFY metadata. Fails on slave zones and zones that do not have DNSSEC.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- 200 OK – OK Returns: string

Objects

A Zone object represents an authoritative DNS Zone.

A Resource Record Set (below as “RRset”) are all records for a given name and type.

Comments are per-RRset.

Zone

This represents an authoritative DNS Zone.

Object Properties

- **id** (*string*) – Opaque zone id (string), assigned by the server, should not be interpreted by the application. Guaranteed to be safe for embedding in URLs.
- **name** (*string*) – Name of the zone (e.g. “example.com.”) MUST have a trailing dot
- **type** (*string*) – Set to “Zone”
- **url** (*string*) – API endpoint for this zone
- **kind** (*string*) – Zone kind, one of “Native”, “Master”, “Slave”
- **rrsets** (*[RRSet]*) – RRSets in this zone
- **serial** (*integer*) – The SOA serial number
- **notified_serial** (*integer*) – The SOA serial notifications have been sent out for
- **masters** (*[string]*) – List of IP addresses configured as a master for this zone (“Slave” type zones only)
- **dnssec** (*boolean*) – Whether or not this zone is DNSSEC signed (inferred from presigned being true XOR presence of at least one cryptokey with active being true)
- **nsec3param** (*string*) – The NSEC3PARAM record
- **nsec3narrow** (*boolean*) – Whether or not the zone uses NSEC3 narrow
- **presigned** (*boolean*) – Whether or not the zone is pre-signed
- **soa_edit** (*string*) – The SOA-EDIT metadata item
- **soa_edit_api** (*string*) – The SOA-EDIT-API metadata item
- **api_rectify** (*boolean*) – Whether or not the zone will be rectified on data changes via the API
- **zone** (*string*) – MAY contain a BIND-style zone file when creating a zone
- **account** (*string*) – MAY be set. Its value is defined by local policy
- **nameservers** (*[string]*) – MAY be sent in client bodies during creation, and MUST NOT be sent by the server. Simple list of strings of nameserver names, including the trailing dot. Not required for slave zones.

RRSet

This represents a Resource Record Set (all records with the same name and type).

Object Properties

- **name** (*string*) – Name for record set (e.g. “www.powerdns.com.”)
- **type** (*string*) – Type of this record (e.g. “A”, “PTR”, “MX”)
- **t1** (*integer*) – DNS TTL of the records, in seconds. MUST NOT be included when changetype is set to “DELETE”.
- **changetype** (*string*) – MUST be added when updating the RRSet. Must be REPLACE or DELETE. With DELETE, all existing RRs matching name and type will be deleted, including all comments. With REPLACE: when records is present, all existing RRs matching name and type will be deleted, and then new records given in records will be created. If no records are left, any existing comments will be deleted as well. When comments is present, all existing comments for the RRs matching name and type will be deleted, and then new comments given in comments will be created.
- **records** (*[Record]*) – All records in this RRSet. When updating Records, this is the list of new records (replacing the old ones). Must be empty when changetype is set to DELETE. An empty list results in deletion of all records (and comments).

- **comments** (*[Comment]*) – List of Comment. Must be empty when changetype is set to DELETE. An empty list results in deletion of all comments. `modified_at` is optional and defaults to the current server time.

Record

The RREntry object represents a single record.

Object Properties

- **content** (*string*) – The content of this record
- **disabled** (*boolean*) – Whether or not this record is disabled
- **set-ptr** (*boolean*) – If set to true, the server will find the matching reverse zone and create a PTR there. Existing PTR records are replaced. If no matching reverse Zone, an error is thrown. Only valid in client bodies, only valid for A and AAAA types. Not returned by the server.

Comment

A comment about an RRSet.

Object Properties

- **content** (*string*) – The actual comment
- **account** (*string*) – Name of an account that added the comment
- **modified_at** (*integer*) – Timestamp of the last change to the comment

Note: Switching `dnssec` to `true` (from `false`) sets up DNSSEC signing based on the other flags, this includes running the equivalent of `secure-zone` and `rectify-zone` (if `api_rectify` is set to `true`). This also applies to newly created zones. If `presigned` is `true`, no DNSSEC changes will be made to the zone or cryptokeys.

Note: `notified_serial`, `serial` MUST NOT be sent in client bodies.

Changes made through the Zones API will always yield valid zone data, as the API will reject records with wrong data.

DNSSEC-enabled zones should be *rectified* after changing the zone data. This can be done by the API automatically after a change when the *API-RECTIFY* metadata is set. When creating or updating a zone, the “`api_rectify`” field of the *ZOne* can be set to `true` to enable this behaviour.

Backends might implement additional features (by coincidence or not). These things are not supported through the API.

When creating a slave zone, it is recommended to not set any of `nameservers`, `rrsets` or `zone`.

15.4.3 Cryptokeys

New in version 4.1.0.

Allows for modifying DNSSEC key material via the API.

Endpoints

GET `/servers/{server_id}/zones/{zone_id}/cryptokeys`

Get all CryptoKeys for a zone, except the privatekey

Parameters

- **server_id** (*string*) – The id of the server to retrieve

- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- **200 OK** – List of Cryptokey objects Returns: array of *Cryptokey* objects

POST /servers/{server_id}/zones/{zone_id}/cryptokeys

Creates a Cryptokey

This method adds a new key to a zone. The key can either be generated or imported by supplying the content parameter. if content, bits and algo are null, a key will be generated based on the default-ksk-algorithm and default-ksk-size settings for a KSK and the default-zsk-algorithm and default-zsk-size options for a ZSK.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –

Status Codes

- **201 Created** – Created Returns: *Cryptokey* object

GET /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}

Returns all data about the CryptoKey, including the privatekey.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve
- **cryptokey_id** (*string*) – The id value of the CryptoKey

Status Codes

- **200 OK** – Cryptokey Returns: *Cryptokey* object

PUT /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}

This method (de)activates a key from zone_name specified by cryptokey_id

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –
- **cryptokey_id** (*string*) – Cryptokey to manipulate

Status Codes

- **204 No Content** – OK
- **422 Unprocessable Entity** – Returned when something is wrong with the content of the request. Contains an error message

DELETE /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}

This method deletes a key specified by cryptokey_id.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve
- **cryptokey_id** (*string*) – The id value of the Cryptokey

Status Codes

- **204 No Content** – OK
- **422 Unprocessable Entity** – Returned when something is wrong with the content of the request. Contains an error message

Objects

Cryptokey

Describes a DNSSEC cryptographic key

Object Properties

- **type** (*string*) – set to “Cryptokey”
- **id** (*string*) – The internal identifier, read only
- **keytype** (*string*) –
- **active** (*boolean*) – Whether or not the key is in active use
- **dnskey** (*string*) – The DNSKEY record for this key
- **ds** (*[string]*) – An array of DS records for this key
- **privatekey** (*string*) – The private key in ISC format
- **algorithm** (*string*) – The name of the algorithm of the key, should be a mnemonic
- **bits** (*integer*) – The size of the key

15.4.4 Metadata

Endpoints

GET /servers/{server_id}/zones/{zone_id}/metadata

Get all the MetaData associated with the zone.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve

Status Codes

- **200 OK** – List of Metadata objects Returns: array of *Metadata* objects

POST /servers/{server_id}/zones/{zone_id}/metadata

Creates a set of metadata entries

Creates a set of metadata entries of given kind for the zone. Existing metadata entries for the zone with the same kind are not overwritten.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –

Status Codes

- **204 No Content** – OK

GET /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}

Get the content of a single kind of domain metadata as a list of MetaData objects.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve
- **metadata_kind** (*string*) – ???

Status Codes

- **200 OK** – List of Metadata objects Returns: *Metadata* object

PUT /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}
Modify the content of a single kind of domain metadata.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –
- **metadata_kind** (*string*) – The kind of metadata

Status Codes

- **204 No Content** – OK

DELETE /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}
Delete all items of a single kind of domain metadata.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve
- **metadata_kind** (*string*) – ???

Status Codes

- **204 No Content** – OK

Objects

Metadata

Represents zone metadata

Object Properties

- **kind** (*string*) – Name of the metadata
- **metadata** (*[string]*) – Array with all values for this metadata kind.

15.4.5 Searching

The API allows searching for data in *Zones*, *Comments* and *RRSets*.

Endpoints

GET /servers/{server_id}/search-data
Search the data inside PowerDNS

Search the data inside PowerDNS for *search_term* and return at most *max_results*. This includes zones, records and comments. The *** character can be used in *search_term* as a wildcard character and the *?* character can be used as a wildcard for a single character.

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Query Parameters

- **q** (*string*) – The string to search for
- **max** (*integer*) – Maximum number of entries to return

Status Codes

- **200 OK** – Returns a JSON array with results Returns: array of *SearchResult* objects

GET /servers/{server_id}/search-log
Query the log, filtered by search_term.

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Query Parameters

- **q** (*string*) – The string to search for

Status Codes

- **200 OK** – Returns a single JSON object with a single array of strings. Returns: string

Objects

SearchResult

Object Properties

- **content** (*string*) –
- **disabled** (*boolean*) –
- **name** (*string*) –
- **object_type** (*string*) – set to one of “record, zone, comment”
- **zone_id** (*string*) –
- **zone** (*string*) –
- **type** (*string*) –
- **ttl** (*integer*) –

15.4.6 Statistics

Endpoints

GET /servers/{server_id}/statistics
Query statistics.

Query PowerDNS internal statistics. Returns a list of BaseStatisticItem derived elements.

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Status Codes

- **200 OK** – List of Statistic Items Returns: array

Objects

MANUAL PAGES

The PowerDNS Authoritative Server comes with many binaries. The manual pages for these programs are included here:

16.1 calidns

16.1.1 Synopsis

calidns [*OPTIONS*] *QUERY_FILE* *DESTINATION* *INITIAL_QPS* *HITRATE*

16.1.2 Description

calidns reads queries from *QUERY_FILE* and sends them as a recursive query to *DESTINATION* (an IPv4 or IPv6 address, optionally with a port number), starting at *INITIAL_QPS* queries per second and aims to have a cache hitrate of *HITRATE* percent.

It will then try to determine the maximum amount of queries per second the recursor can handle with the aforementioned *HITRATE*.

16.1.3 QUERY_FILE format

The format of the *QUERY_FILE* is very simple, it should contain “QNAME QTYPE” tuples, one per line. For example:

```
powerdns.com A
powerdns.com AAAA
google.com A
```

This is similar to Alexa top 1 million list.

16.1.4 Options

- increment <NUM>** On every subsequent run, multiply the number of queries per second by *NUM*. By default, this is 1.1.
- want-recursion** Set this flag to send queries with the Recursion Desired flag set.

16.2 dnshulktest

16.2.1 Synopsis

dnshulktest [*OPTION*]. . . *IPADDRESS PORT [LIMIT]*

16.2.2 Description

dnshulktest sends a large amount of different queries (for up to *LIMIT* different domains) to the nameserver at *IPADDRESS* on port *PORT*. It reads the domain names from STDIN in the alexa topX format and outputs statistics on STDOUT.

16.2.3 Options

--help, -h	Show a summary of options.
--quiet, -q	Don't show information on individual queries.
--type, -t <TYPE>	Query the nameserver for <i>TYPE</i> , A by default.
--envoutput, -e	Write results on STDOUT as shell environment variables
--version	Display the version of dnshulktest

16.3 dnsggram

16.3.1 Synopsis

dnsggram *INFILE* . . .

16.3.2 Description

dnsggram takes one or more *INFILE*s in PCAP format and generates statistics on 5 second segments allowing the study of intermittent resolver issues.

16.3.3 Options

None

16.3.4 See also

pcap(3PCAP), tcpdump(8)

16.4 dnspcap2protobuf

16.4.1 Synopsis

dnspcap2protobuf *PCAPFILE OUTFILE*

16.4.2 Description

dnscap2protobuf reads the PCAP file *PCAPFILE* for DNS queries and responses and writes these in the PowerDNS protobuf format to *OUTFILE*.

16.4.3 Options

--help	Show a summary of options.
--version	Display the version of dnscap2protobuf

16.5 dnsreplay

16.5.1 Synopsis

dnsreplay [*OPTION*]... *FILENAME ADDRESS [PORT]*

16.5.2 Description

This program takes recorded questions and answers and replays them to the specified nameserver and reporting afterwards which percentage of answers matched, were worse or better.

dnsreplay compares the answers and some other metrics with the actual ones with those found in the dumpfile.

By default it only replay queries with recursion-desired flag set.

16.5.3 Options

FILENAME is expected to be an PCAP file. The queries are send to the DNS server specified as *ADDRESS* and *PORT*.

ADDRESS IPv4 or IPv6 address of the nameserver to replay *FILENAME* to.

PORT if omitted, 53 will be used.

--help, -h	Show summary of options.
--ecs-mask <VAL>	When EDNS forwarding an IP address, mask out first octet with this value
--ecs-stamp <FLAG>	Add original IP address as EDNS Client Subnet Option when forwarding to reference server
--pcap-dns-port <VAL>	Look at packets from or to this port in the PCAP. Default is 53.
--packet-limit <NUM>	Stop after replaying <i>NUM</i> packets. Default for <i>NUM</i> is 0, which means no limit.
--quiet <FLAG>	If <i>FLAG</i> is set to 1. dnsreplay will not be very noisy with its output. This is the default.
--recursive <FLAG>	If <i>FLAG</i> is set to 1. dnsreplay will only replay queries with recursion desired flag set. This is the default.
--speedup <FACTOR>	Replay queries with this speedup <i>FACTOR</i> . Default is 1.
--timeout-msec <MSEC>	Wait at least <i>MSEC</i> milliseconds for a reply. Default is 500.

16.5.4 Bugs

dnsreplay has no certain handling for timeouts. It handles around at most 65536 outstanding answers.

16.5.5 See also

pcap(3PCAP), tcpdump(8), dnswasher(1)

16.6 dnsscan

16.6.1 Synopsis

dnsscan *INFILE*...

16.6.2 Description

dnsscan takes one or more *INFILE*s in PCAP format and generates a list of the number of queries per qtype.

16.6.3 Options

None

16.6.4 See also

pcap(3PCAP), tcpdump(8)

16.7 dnsscope

16.7.1 Synopsis

dnsscope [*OPTION*]... *INFILE*

16.7.2 Description

dnsscope takes an *INFILE* in PCAP format. It generates some simple statistics outputs these to STDOUT.

16.7.3 Options

INFILE Path to a PCAP file.

-h, --help	Show the help.
--rd	Only process packets in <i>INFILE</i> with the RD (Recursion Desired) flag set. By default, we process all DNS packets in <i>INFILE</i> .
--ipv4=<state>	Process IPv4 packets. On by default, disable with --ipv4 false .
--ipv6=<state>	Process IPv6 packets. On by default, disable with --ipv6 false .
--servfail-tree	Figure out subtrees that generate servfails.
-l, --load-stats	Emit per-second load statistics (questions, answers, outstanding).
-w <file>, --write-failures <file>	Write weird packets to a PCAP file at <i>FILENAME</i> .
-v, --verbose	Be more verbose.

16.7.4 See also

pcap(3PCAP), tcpdump(8)

16.8 dnstcpbench

16.8.1 Synopsis

dnstcpbench [*OPTION*]... *REMOTE-ADDRESS* [*REMOTE-PORT*]

16.8.2 Description

dnstcpbench reads DNS queries (by default from standard input) and sends them out in parallel to a remote nameserver. By default TCP/IP is used, but optionally, UDP is tried first, which allows for the benchmarking of TCP/IP fallback.

The program reports both mean and median numbers for queries per second and UDP and TCP latency. Each query only counts once, even if it is tried over UDP first. This effectively means that passing ‘-u’ can lower query rates if many queries get shunted to TCP.

The input format is one query per line: qname single-space qtype. An example:

```
www.powerdns.com ANY
```

When benchmarking extended runs, it may be necessary to enable TIME_WAIT recycling, as TCP/IP port tuples may otherwise run out. On Linux this is performed by running:

```
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
```

The equivalent for IPv6 is not known.

16.8.3 Options

-f, <FILENAME>, -file <FILENAME> *FILENAME* from which to read queries. Defaults to standard input if unspecified. -h, -help Provide a helpful message. -timeout-msec <MSEC> *MSEC* milliseconds to wait for an answer. -u, -udp-first Attempt resolution via UDP first, only do TCP if truncated answer is received. -v, -verbose Be wordy on what the program is doing. -workers <NUM> Use *NUM* parallel worker threads.

REMOTE-ADDRESS: IPv4 or IPv6 to test against.

REMOTE-PORT: Port to test against, defaults to 53.

16.8.4 Bugs

Currently the timeout code does not actually perform non-blocking connects or writes. So a slow connect or slow writes will still cause low performance and delays.

Median queries per second statistics are reported as 0 for sub-second runs.

16.9 dnswasher

16.9.1 Synopsis

dnswasher *INFILE* [*INFILE*] *OUTFILE*

16.9.2 Description

dnswasher takes one or more *INFILE*s in PCAP format and writes out *OUTFILE* also in PCAP format, while obfuscating end-user IP addresses.

This is useful to share data with third parties while attempting to protect the privacy of your users.

The INFILEs must be of identical PCAP type.

Please check the output of **dnswasher** to make sure no customer IP addresses remain. Also realize that sufficient data could allow individuals to be re-identified based on the domain names they care about.

16.9.3 Options

None

16.9.4 See also

pcap(3PCAP), tcpdump(8)

16.10 dumresp

16.10.1 Synopsis

dumresp *LOCAL-ADDRESS LOCAL-PORT NUMBER-OF-PROCESSES*

16.10.2 Description

dumresp accepts DNS packets on *LOCAL-ADDRESS:LOCAL-PORT* and simply replies with the same query, with the QR bit set. When *NUMBER-OF-PROCESSES* is set to anything but 1, **dumresp** will spawn *NUMBER-OF-PROCESSES* forks and use the SO_REUSEPORT option to bind to the port.

16.10.3 Options

None

16.10.4 See also

socket(7)

16.11 ixfrdist

16.11.1 Synopsis

ixfrdist [*OPTION*]... *DOMAIN* [*DOMAIN*]...

16.11.2 Description

ixfrdist transfers zones from an authoritative server and re-serves these zones over AXFR and IXFR. It checks the SOA serial for all *DOMAINs* from the server set with **–server-address** and downloads new versions to **–work-dir**. This working directory has the following structure: `work-dir/ZONE/SERIAL`, e.g. `work-dir/rpz.example./2018011902`.

When a SOA query comes in on the address(es) set with **–listen-address**, **ixfrdist** responds with the latest SOA for the zone it has. This query can be followed up with an IXFR or AXFR query, which will then be served. Should an IXFR be served, **ixfrdist** will condense the diff into the IXFR.

When using **–uid** or **–gid** the **–work-dir** directory will be accessed (and potentially created) as the proved user/group.

16.11.3 Options

--help	Show all supported options
--verbose	Log INFO messages
--debug	Log INFO and DEBUG messages
--version	Display the version of ixfrdist
--listen-address <ADDRESS>	Listen on <i>ADDRESS</i> for incoming queries. <i>ADDRESS</i> may contain a port number, when unset 53 is assumed. This option can be given multiple times. When not set, 127.0.0.1:53 is assumed.
--server-address <ADDRESS>	IP address and port of the upstream server. 127.0.0.1:5300 by default.
--work-dir <DIR>	Path to a directory where the AXFR data are saved. By default, this is the current working directory.
--keep <NUM>	Keep at most <i>NUM</i> versions of any zone. By default, 20 versions are kept.
--uid <UID>	Drop effective user-id to <i>UID</i> after binding the listen sockets
--gid <GID>	Drop effective group-id to <i>GID</i> after binding the listen sockets

16.11.4 See also

`ixplore(1)`, `pdns_server(1)`

16.12 ixplore

16.12.1 Synopsis

```
ixplore COMMAND COMMAND_OPT...
ixplore diff ZONE BEFORE AFTER
ixplore track IP ADDRESS PORT ZONE DIRECTORY
```

16.12.2 Description

ixplore is a tool to work with IXFR (incremental zonetransfers) in two modes (specified by *COMMAND*): diff or track.

In the ‘diff’ mode, it will show a diff(1)-like output between *BEFORE* and *AFTER*.

In the ‘track’ mode, **ixplore** consumes IXFRs from *IP ADDRESS* and writes the resulting zonefiles out to *DIRECTORY/ZONE*-serial. If no initial zonefiles exist, an initial AXFR will be done first. **ixplore** will then check the SOA serial on *IP ADDRESS* for *ZONE* every SOA Refresh seconds and perform an IXFR if the serial has increased.

16.12.3 Options

16.12.4 diff-mode

ZONE The name of the zone the IXFRs are consumed from.

BEFORE Path to the ‘before’ zonefile.

AFYER Path to the ‘after’ zonefile.

16.12.5 track-mode

IP ADDRESS The IP address to consume IXFRs from.

PORT The port to use on *IP ADDRESS*.

ZONE Name of the zone to track changes of.

DIRECTORY Directory where the zonefiles will be stored.

16.12.6 See also

diff(1)

16.13 nproxy

nproxy - DNS notification proxy

16.13.1 Synopsis

nproxy -powerdns-address *ADDRESS* [*OPTION*]. . . *ADDRESS*. . .

16.13.2 Description

nproxy is a simple daemon that reads DNS NOTIFY queries on one address and forwards them to an ‘inner’ nameserver that will process the notification.

Its usecase is e.g. a private authoritative server inside a NAT or firewalled LAN where **nproxy** is deployed in the DMZ.

The PowerDNS Authoritative Server has the trusted-notification-proxy option that should be set to the address set with *-origin-address* to accept these proxied notifications.

nproxy also has a health-check option built in. A query for ‘pdns.nproxy.’ with QType ‘TXT’ will be responded to with an answer of “OK” (inside the TXT record. When the query is for an A-record, ‘1.2.3.4.’ is returned.

16.13.3 Options

- powerdns-address** <ADDRESS> IP address of the PowerDNS server to forward the notifications to.
- chroot** <PATH> chroot to *PATH* for additional security.
- setuid** <UID> setuid to this numerical *UID*.
- setgid** <GID> setgid to this numerical *GID*.
- origin-address** <ADDRESS> Set the source of the notifications sent to PowerDNS to *ADDRESS*. By default, the best matching address (kernel's choice) is used.
- listen-address** <ADDRESS> IP addresses to listen on.
- listen-port** <PORT> Source port to listen on, 53 by default.
- d, --daemon** <ARG> Set *ARG* to 0 to disable running in the background.
- v, --verbose** Be verbose

16.14 nsec3dig

16.14.1 Synopsis

nsec3dig *IPADDRESS PORT QNAME QTYPE* [recurse]

16.14.2 Description

nsec3dig sends a query for *QNAME* and *QTYPE* to the nameserver at *IPADDRESS* on port *PORT* and prints whether and why the NSEC3 proofs are correct. Using the 'recurse' option sets the Recursion Desired (RD) bit in the query.

16.14.3 Example

```
nsec3dig 8.8.8.8 53 doesntexist.isoc.nl TXT recurse
```

16.15 pdns_control

16.15.1 Synopsis

pdns_control [*OPTION*]... *COMMAND*

16.15.2 Description

pdns_control is used to send commands to a running PowerDNS nameserver.

16.15.3 Options

- help** Show summary of options.
- chroot=<DIR>** Directory where PowerDNS is chrooted.
- config-dir=<DIR>** Location of configuration directory (pdns.conf).

--config-name=<NAME> Name of this virtual configuration - will rename the binary image.
--remote-address=<ADDRESS> Remote address to query.
--remote-port=<PORT> Remote port to query.
--secret=<SECRET> Secret needed to connect to remote PowerDNS.
--socket-dir=<DIR> Where the controlsocket lives.

16.15.4 Commands

bind-add-zone *DOMAIN FILENAME*

When using the bindbackend, add a zone. This zone is added in-memory and served immediately. Note that this does not add the zone to the bind-config file. *FILENAME* must be an absolute path.

bind-domain-status [*DOMAIN...*]

When using the bindbackend, list status of all domains. Optionally, append *DOMAIN*s to get the status of specific zones.

bind-list-rejects

When using the bindbackend, get a list of all rejected domains.

bind-reload-now *DOMAIN [DOMAIN...]*

When using the bindbackend, immediately reload *DOMAIN* from disk.

ccounts

Show the content of the cache.

current-config

Show the currently running configuration. The output has the same format as `pdns_server --config`. You'll notice that all the are uncommented. This is because PowerDNS simply has values, and the default isn't known at runtime.

cycle

Restart the nameserver so it reloads its configuration. Only works when the server is running in guardian mode.

list

Dump all variables and their values in a comma separated list, equivalent to `show *`.

list-zones [*master,slave,native*]

Show a list of zones, optionally filter on the type of zones to show.

notify *DOMAIN*

Adds *DOMAIN* to the notification list, causing PowerDNS 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. Use * to notify for all domains. (Note that you may need to escape the * sign in your shell.)

notify-host *DOMAIN ADDRESS*

Same as above but with operator specified IP *ADDRESS* as destination, to be used if you know better than PowerDNS.

ping, rping

Check if the server is still alive. Will return 'PONG' when it is. `ping` works when running inside a guardian, whereas `rping` works when running without a guardian.

purge [*RECORD*]

Purge entries from the cache. If *RECORD* ends with a dollar (\$) all entries that end with that name are removed. If no record is specified the entire cache is purged.

qtypes

Get a count of queries per qtype on standard out.

quit

Tell a running `pdns_server` to quit.

rediscover

Instructs backends that new domains may have appeared in the database, or, in the case of the Bind backend, in `named.conf`.

reload

Instruct the server to reload all its zones, this will not add new zones.

remotes

Get the top number of remote addresses (clients).

respsizes

Get a histogram of the response sizes.

retrieve *DOMAIN*

Retrieve slave *DOMAIN* from its master. Done nearly immediately.

set *VARIABLE VALUE*

Set the configuration parameter *VARIABLE* to *VALUE*. Currently only the query-logging can be set.

show *VARIABLE*

Show a single statistic, as present in the output of the list command.

status

Show usage statistics. This only works if the server is running in guardian mode.

token-login *MODULE SLOT PIN*

Log on to a PKCS#11 slot. You only need to login once per slot, even if you have multiple keys on single slot. Only available if PowerDNS was compiled with PKCS#11 support.

uptime

Show the uptime of the running server.

version

Print the version of the running pdns daemon.

16.15.5 See also

pdns_server(1)

16.16 pdns_notify

16.16.1 Synopsis

pdns_notify *IP_ADDRESS[:PORT] DOMAIN*

16.16.2 Description

pdns_notify sends a DNS NOTIFY message to *IP_ADDRESS*, by default on port 53, for *DOMAIN* and prints the remote nameserver's response.

16.16.3 Options

None

16.17 pdns_server

16.17.1 Synopsis

pdns_server [*OPTION*]

16.17.2 Description

The PowerDNS Authoritative Server 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. Please see the online documentation for more information.

16.17.3 Options

See the online documentation for all options

--daemon	Indicate if the server should run in the background as a real daemon, or in the foreground.
--guardian	Run pdns_server inside a guardian. This guardian monitors the performance of the inner pdns_server instance. It is also this guardian that pdns_control talks to.
--control-console	Run the server in a special monitor mode. This enables detailed logging and exposes the raw control socket.
--loglevel=<LEVEL>	Set the logging level.
--help	To view more options that are available use this program.

16.17.4 See also

pdns_control(1), pdnsutil(1), <https://doc.powerdns.com/>

16.18 pdnsutil

16.18.1 Synopsis

pdnsutil [OPTION]... *COMMAND*

16.18.2 Description

pdnsutil (formerly **pdnssec**) is a powerful command that is the operator-friendly gateway into DNSSEC and zone management for PowerDNS. Behind the scenes, **pdnsutil** manipulates a PowerDNS backend database, which also means that for many databases, **pdnsutil** can be run remotely, and can configure key material on different servers.

16.18.3 Options

-h, --help	Show summary of options
-v, --verbose	Be more verbose.
--force	Force an action
--config-name <NAME>	Virtual configuration name
--config-dir <DIR>	Location of pdns.conf. Default is /etc/powerdns.

16.18.4 COMMANDS

There are many available commands, this section splits them up into their respective uses

16.18.5 DNSSEC RELATED COMMANDS

Several commands manipulate the DNSSEC keys and options for zones. Some of these commands require an *ALGORITHM* to be set. The following algorithms are supported:

- rsasha1
- rsasha256
- rsasha512
- gost
- ecdsa256
- ecdsa384

activate-zone-key *ZONE KEY-ID* Activate a key with id *KEY-ID* within a zone called *ZONE*.

add-zone-key *ZONE* {*KSK,ZSK*} [*active,inactive*] *KEYBITS ALGORITHM* Create a new key for zone *ZONE*, and make it a KSK or a ZSK, with the specified algorithm. The key is inactive by default, set it to **active** to immediately use it to sign *ZONE*. Prints the id of the added key.

create-bind-db *FILE* Create DNSSEC database (sqlite3) at *FILE* for the BIND backend. Remember to set `bind-dnssec-db=*FILE*` in your `pdns.conf`.

deactivate-zone-key *ZONE KEY-ID* Deactivate a key with id *KEY-ID* within a zone called *ZONE*.

disable-dnssec *ZONE* Deactivate all keys and unset *PRESIGNED* in *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.

generate-zone-key {*KSK,ZSK*} [*ALGORITHM*] [*KEYBITS*] Generate a ZSK or KSK to stdout with specified algorithm and bits and print it on STDOUT. If *ALGORITHM* is not set, RSASHA512 is used. If *KEYBITS* is not set, an appropriate keysize is selected for *ALGORITHM*.

import-zone-key *ZONE FILE* {*KSK,ZSK*} Import from *FILE* 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. Prints the id of the added key.

remove-zone-key *ZONE KEY-ID* Remove a key with id *KEY-ID* from a zone called *ZONE*.

set-nsec3 *ZONE* '*HASH-ALGORITHM FLAGS ITERATIONS SALT*' [*narrow*] Sets NSEC3 parameters for this zone. The quoted parameters are 4 values that are used for the the NSEC3PARAM record and decide how NSEC3 records are created. The NSEC3 parameters must be quoted on the command line. *HASH-ALGORITHM* must be 1 (SHA-1). Setting *FLAGS* to 1 enables NSEC3 opt-out operation. Only do this if you know you need it. For *ITERATIONS*, please consult RFC 5155, section 10.3. And be aware that a high number might overload validating resolvers. The *SALT* is a hexadecimal string encoding the bits for the salt, or - to use no salt. Setting **narrow** will make PowerDNS send out “white lies” about the next secure record. Instead of looking it up in the database, it will send out the hash + 1 as the next secure record. A sample commandline is: “`pdnsutil set-nsec3 powerdnssec.org '1 1 1 ab' narrow`”. **WARNING:** If running in RSASHA1 mode (algorithm 5 or 7), switching from NSEC to NSEC3 will require a DS update in the parent zone.

unset-nsec3 *ZONE* Converts *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!

set-publish-cds *ZONE* [*DIGESTALGOS*] Set *ZONE* to respond to queries for its CDS records. the optional argument *DIGESTALGOS* should be a comma-separated list of DS algorithms to use. By default, this is 1,2 (SHA1 and SHA2-256).

set-publish-cdnskey *ZONE* Set *ZONE* to publish CDNSKEY records.

unset-publish-cds *ZONE* Set *ZONE* to stop responding to queries for its CDS records.

unset-publish-cdnskey *ZONE* Set *ZONE* to stop publishing CDNSKEY records.

16.18.6 TSIG RELATED COMMANDS

These commands manipulate TSIG key information in the database. Some commands require an *ALGORITHM*, the following are available:

- `hmac-md5`
- `hmac-sha1`
- `hmac-sha224`
- `hmac-sha256`
- `hmac-sha384`
- `hmac-sha512`

activate-tsig-key *ZONE NAME {master,slave}* Enable TSIG authenticated AXFR using the key *NAME* for zone *ZONE*. This sets the `TSIG-ALLOW-AXFR` (master) or `AXFR-MASTER-TSIG` (slave) zone metadata.

deactivate-tsig-key *ZONE NAME {master,slave}* Disable TSIG authenticated AXFR using the key *NAME* for zone *ZONE*.

delete-tsig-key *NAME* Delete the TSIG key *NAME*. Warning, this does not deactivate said key.

generate-tsig-key *NAME ALGORITHM* Generate new TSIG key with name *NAME* and the specified algorithm.

import-tsig-key *NAME ALGORITHM KEY* Import *KEY* of the specified algorithm as *NAME*.

list-tsig-keys Show a list of all configured TSIG keys.

16.18.7 ZONE MANIPULATION COMMANDS

add-record *ZONE NAME TYPE [TTL] CONTENT* Add one or more records of *NAME* and *TYPE* to *ZONE* with *CONTENT* and optional *TTL*. If *TTL* is not set, default will be used.

create-zone *ZONE* Create an empty zone named *ZONE*.

create-slave-zone *ZONE MASTER [MASTER]..* Create a new slave zone *ZONE* with masters *MASTER*. All *MASTERS* need to be IP addresses with an optional port.

change-slave-zone-master *ZONE MASTER [MASTER]..* Change the masters for slave zone *ZONE* to new masters *MASTER*. All *MASTERS* need to be IP addresses with an optional port.

check-all-zones Check all zones for correctness.

check-zone *ZONE* Check zone *ZONE* for correctness.

clear-zone *ZONE* Clear the records in zone *ZONE*, but leave actual domain and settings unchanged

delete-zone *ZONE:* Delete the zone named *ZONE*.

edit-zone *ZONE* Opens *ZONE* in zonefile format (regardless of backend it was loaded from) in the editor set in the environment variable **EDITOR**. if **EDITOR** is empty, *pdnsutil* falls back to using *editor*.

get-meta *ZONE [ATTRIBUTE]...* Get zone metadata. If no *ATTRIBUTE* given, lists all known.

hash-zone-record *ZONE RNAME* This convenience command hashes the name *RNAME* according to the NSEC3 settings of *ZONE*. Refuses to hash for zones with no NSEC3 settings.

list-keys [*ZONE*] List DNSSEC information for all keys or for *ZONE*.

list-all-zones: List all zone names.

list-zone *ZONE* Show all records for *ZONE*.

load-zone *ZONE FILE* Load records for *ZONE* from *FILE*. If *ZONE* already exists, all records are overwritten, this operation is atomic. If *ZONE* doesn't exist, it is created.

rectify-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.

rectify-all-zones Calculates the 'ordname' and 'auth' fields for all zones so they comply with DNSSEC settings. Can be used to fix up migrated data. Can always safely be run, it does no harm.

secure-zone *ZONE* Configures a zone called *ZONE* with reasonable DNSSEC settings. You should manually run 'pdnsutil rectify-zone' afterwards.

secure-all-zones [**increase-serial**] Configures all zones that are not currently signed with reasonable DNSSEC settings. Setting **increase-serial** will increase the serial of those zones too. You should manually run 'pdnsutil rectify-all-zones' afterwards.

set-kind *ZONE KIND* Change the kind of *ZONE* to *KIND* (master, slave, native).

set-account *ZONE ACCOUNT* Change the account (owner) of *ZONE* to *ACCOUNT*.

add-meta *ZONE ATTRIBUTE VALUE [VALUE]...* Append *VALUE* to the existing *ATTRIBUTE* metadata for *ZONE*. Will return an error if *ATTRIBUTE* does not support multiple values, use **set-meta** for these values.

set-meta *ZONE ATTRIBUTE [VALUE]...* Set domainmetadata *ATTRIBUTE* for *ZONE* to *VALUE*. An empty value clears it.

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*.

test-schema *ZONE* Test database schema, this creates the zone *ZONE*

unset-presigned *ZONE* Disables presigned operation for *ZONE*.

16.18.8 DEBUGGING TOOLS

backend-cmd *BACKEND CMD [CMD..]* Send a text command to a backend for execution. GSQL backends will take SQL commands, other backends may take different things. Be careful!

bench-db [*FILE*] Perform a benchmark of the backend-database. *FILE* can be a file with a list, one per line, of domain names to use for this. If *FILE* is not specified, powerdns.com is used.

16.18.9 See also

pdns_server (1), pdns_control (1)

16.19 saxfr

16.19.1 Synopsis

saxfr *IPADDRESS PORT ZONE [Options]*

16.19.2 Description

saxfr does a zone-transfer (AXFR) of *ZONE* from the nameserver at *IPADDRESS* on port *PORT* and displays the transferred zone with NSEC3 information truncated. See below how to show this information.

16.19.3 Options

showdetails Show all the data in the NSEC3 and DNSKEY RDATA.

showflags Show the NSEC3 flags in the RDATA.

unhash Unhash the NSEC3 names to the normal names.

16.20 sdig

16.20.1 Synopsis

sdig *IPADDRESS PORT QNAME QTYPE [OPTION]*

16.20.2 Description

sdig sends a DNS query to *IPADDRESS* on port *PORT* and displays the answer in a formatted way.

16.20.3 Options

These options can be added to the commandline in any order. **dnssec** : Set the DO bit to request DNSSEC information.

hidesoadetails Don't show the SOA serial in the response.

recurse Set the RD bit in the question.

showflags Show the NSEC3 flags in the response.

tcp Use TCP instead of UDP to send the query.

ednssubnet *SUBNET* Send *SUBNET* in the edns-client-subnet option. If this option is not set, no edns-client-subnet option is set in the query.

16.21 zone2json

16.21.1 Synopsis

zone2json {**--named-conf**=*PATH*, **--zone-file**=*PATH* [**--zone-name**=*NAME*]} [*OPTION*]

16.21.2 Description

zone2json parses Bind named.conf files and zonefiles and outputs JSON on standard out, which can then be fed to the PowerDNS API.

zone2json understands the Bind master file extension \$GENERATE and will also honour \$ORIGIN and \$TTL.

16.21.3 Options

16.21.4 INPUT Options

--named-conf=<*PATH*> Read *PATH* to get the bind configuration

--zone=<*PATH*> Parse only the zone file at *PATH* Conflicts with **--named-conf** parameter.

--zone-name=<NAME> When parsing a single zone without \$ORIGIN statement, set *ZONE* as the zone name.

16.21.5 OTHER Options

--help List all options
--on-error-resume-next Ignore missing zone files during parsing. Dangerous.
--verbose Be verbose during conversion.

16.21.6 See also

pdns_server(1)

16.22 zone2ldap

16.22.1 Synopsis

zone2ldap { **-named-conf=PATH**, **-zone-file=PATH** **-zone-name=NAME** } [*OPTION*]. . .

16.22.2 Description

zone2ldap is a program that converts bind zonefiles to ldif format which can be inserted into an LDAP server.

16.22.3 Options

--help Show summary of options.
--basedn=<DN> Base DN to store objects below
--dnsttl Add dnsttl attribute to every entry
--layout=<layout> How to arrange entries in the directory (“simple” or “tree”)
--named-conf=<PATH> Path to a Bind named.conf to parse
--resume Continue after errors
--verbose Verbose comments on operation
--zone-file=<PATH> Zone file to parse
--zone-name=<NAME> Specify a zone name if zone is set

16.22.4 See also

pdns_server(1)

16.23 zone2sql

16.23.1 Synopsis

zone2sql { **-named-conf=PATH**, **-zone-file=PATH** [**-zone-name=NAME**] } [*Options*]

16.23.2 Description

zone2sql parses Bind named.conf files and zonefiles and outputs SQL on standard out, which can then be fed to your database.

zone2sql understands the Bind master file extension \$GENERATE and will also honour \$ORIGIN and \$TTL.

For backends supporting slave operation there is also an option to keep slave zones as slaves, and not convert them to native operation.

zone2sql can generate SQL for the Generic MySQL, Generic PostgreSQL, Generic SQLite3 and Generic Oracle backends.

16.23.3 Options

16.23.4 INPUT Options

- named-conf=<PATH>** Read *PATH* to get the bind configuration
- zone=<PATH>** Parse only the zone file at *PATH* Conflicts with **--named-conf** parameter.
- zone-name=<NAME>** When parsing a single zone without \$ORIGIN statement, set *ZONE* as the zone name.

16.23.5 BACKENDS

- mysql** Output in format suitable for the default configuration of the Generic MySQL backend.
- pgsql** Output in format suitable for the default configuration of the Generic PostgreSQL backend.
- sqlite** Output in format suitable for the default configuration of the Generic SQLite3 backend.
- oracle** Output in format suitable for the default configuration of the Generic Oracle backend.
- mydns** Output in a format suitable for the MyDNS backend.
- oracle** Output in format suitable for the default configuration of the Oracle backend.

16.23.6 OUTPUT Options

- json-comments** Parse JSON in zonefile comments to set the 'disabled' and 'comment' fields in the database. See *JSON COMMENTS* for more information.
- transactions** If the target SQL backend supports transactions, wrap every domain into a transaction for integrity and possibly higher speed.

16.23.7 OTHER Options

- filter-duplicate-soa** If there's more than one SOA record in the zone (possibly because it was AXFR'd), ignore it. If this option is not set, all SOA records in the zone are emitted.
- help** List all options
- on-error-resume-next** Ignore missing zone files during parsing. Dangerous.

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

16.23.8 JSON COMMENTS

The Generic SQL backends have the ‘comment’ and ‘disabled’ fields in the ‘records’ table. The ‘comment’ field contains a comment for this record (if any) and the ‘disabled’ field tells PowerDNS if the record can be served to clients.

When a zonefile contains a comment like `; json={"comment": "Something", "disabled": true}` and **--json-comments** is provided, the ‘comment’ field will contain “Something” and the ‘disabled’ field will be set to the database’s native true value.

WARNING: Using JSON comments to disable records means that the zone in PowerDNS is different from the one served by BIND, as BIND does not handle the disabled status in the comment.

16.23.9 See also

pdns_server(1)

AUTHORITATIVE SERVER SETTINGS

All PowerDNS Authoritative Server settings are listed here, excluding those that originate from backends, which are documented in the relevant chapters. These settings can be set inside `pdns.conf` or on the commandline when invoking the `pdns` binary.

You can use `+=` syntax to set some variables incrementally, but this requires you to have at least one non-incremental setting for the variable to act as base setting. This is mostly useful for *include-dir* directive.

For boolean settings, specifying the name of the setting without a value means *yes*.

17.1 8bit-dns

- Allow 8 bit dns queries
- Default: no

New in version 4.0.0.

Allow 8 bit DNS queries.

17.2 allow-axfr-ips

- IP ranges, separated by commas
- Default: 127.0.0.0/8,::1

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

17.3 allow-dnsupdate-from

- IP ranges, separated by commas
- Default: 127.0.0.0/8,::1

Allow DNS updates from these IP ranges. Set to empty string to honour `ALLOW-DNSUPDATE-FROM` in *ALLOW-DNSUPDATE-FROM*.

17.4 allow-notify-from

- IP ranges, separated by commas
- Default: 0.0.0.0/0,::/0

Allow AXFR NOTIFY from these IP ranges. Setting this to an empty string will drop all incoming notifies.

17.5 `allow-unsigned-notify`

- Boolean
- Default: yes

New in version 4.0.0.

Turning this off requires all notifications that are received to be signed by valid TSIG signature for the zone.

17.6 `allow-unsigned-supermaster`

- Boolean
- Default: yes

New in version 4.0.0.

Turning this off requires all supermaster notifications to be signed by valid TSIG signature. It will accept any existing key on slave.

17.7 `allow-recursion`

- IP ranges, separated by commas
- Default: 0.0.0.0/0
- Removed in: 4.1.0

By specifying `allow-recursion`, recursion can be restricted to netmasks specified. The default is to allow recursion from everywhere. Example: `allow-recursion=198.51.100.0/24, 10.0.0.0/8, 192.0.2.4`.

17.8 `also-notify`

- IP addresses, separated by commas

When notifying a domain, also notify these nameservers. Example: `also-notify=192.0.2.1, 203.0.113.167`. The IP addresses listed in `also-notify` always receive a notification. Even if they do not match the list in *only-notify*.

17.9 `any-to-tcp`

- Boolean
- Default: yes

Changed in version 4.0.1,: was 'no' before.

Answer questions for the ANY on UDP with a truncated packet that refers the remote server to TCP. Useful for mitigating reflection attacks.

17.10 `api`

- Boolean
- Default: no

Enable/disable the *Built-in Webserver and HTTP API*.

17.11 `api-key`

- String

New in version 4.0.0.

Static pre-shared authentication key for access to the REST API.

17.12 `api-readonly`

- Boolean
- Default: no

New in version 4.0.0.

Disallow data modification through the REST API when set.

17.13 `axfr-lower-serial`

- Boolean
- Default: no

New in version 4.0.4.

Also AXFR a zone from a master with a lower serial.

17.14 `cache-ttl`

- Integer
- Default: 20

Seconds to store packets in the *Packet Cache*.

17.15 `carbon-ourname`

- String
- Default: the hostname of the server

If sending carbon updates, if set, this will override our hostname. Be careful not to include any dots in this setting, unless you know what you are doing. See *Sending metrics to Graphite/Metronome over Carbon*

17.16 carbon-server

- IP Address

Send all available metrics to this server via the carbon protocol, which is used by graphite and metronome. It has to be an address (no hostnames). You may specify an alternate port by appending :port, ex: 127.0.0.1:2004. See *Sending metrics to Graphite/Metronome over Carbon*.

17.17 carbon-interval

- Integer
- Default: 30

If sending carbon updates, this is the interval between them in seconds. See *Sending metrics to Graphite/Metronome over Carbon*.

17.18 chroot

- Path

If set, chroot to this directory for more security. See *Security of PowerDNS*.

Make sure that `/dev/log` is available from within the chroot. Logging will silently fail over time otherwise (on logrotate).

When setting `chroot`, all other paths in the config (except for *config-dir* and *module-dir*) set in the configuration are relative to the new root.

When running on a system where systemd manages services, `chroot` does not work out of the box, as PowerDNS cannot use the `NOTIFY_SOCKET`. Either don't `chroot` on these systems or set the 'Type' of the this service to 'simple' instead of 'notify' (refer to the systemd documentation on how to modify unit-files)

17.19 config-dir

- Path

Location of configuration directory (`pdns.conf`). Usually `/etc/powerdns`, but this depends on `SYSCONFDIR` during compile-time.

17.20 config-name

- String

Name of this virtual configuration - will rename the binary image. See *Running Virtual Instances*.

17.21 control-console

Debugging switch - don't use.

17.22 daemon

- Boolean
- Default: no

Operate as a daemon.

17.23 default-ksk-algorithm

- String
- Default: ecdsa256

Changed in version 4.1.0: Renamed from `default-ksk-algorithms`. No longer supports multiple algorithm names.

The algorithm that should be used for the KSK when running *pdnsutil secure-zone* or using the *Zone API endpoint* to enable DNSSEC. Must be one of:

- rsamd5
- dh
- dsa
- ecc
- rsasha1
- rsasha256
- rsasha512
- ecc-gost
- ecdsa256 (ECDSA P-256 with SHA256)
- ecdsa384 (ECDSA P-384 with SHA384)
- ed25519

Note: Actual supported algorithms depend on the crypto-libraries PowerDNS was compiled against. To check the supported DNSSEC algorithms in your build of PowerDNS, run `pdnsutil list-algorithms`.

17.24 default-ksk-size

- Integer
- Default: whichever is default for *default-ksk-algorithm*

The default keysize for the KSK generated with *pdnsutil secure-zone*. Only relevant for algorithms with non-fixed keysizes (like RSA).

17.25 default-soa-name

- String
- Default: a.misconfigured.powerdns.server

Name to insert in the SOA record if none set in the backend.

17.26 default-soa-edit

- String
- Default: empty

Use this soa-edit value for all zones if no *SOA-EDIT* metadata value is set.

17.27 default-soa-edit-signed

- String
- Default: empty

Use this soa-edit value for all signed zones if no *SOA-EDIT* metadata value is set. Overrides *default-soa-edit*

17.28 default-soa-mail

- String

Mail address to insert in the SOA record if none set in the backend.

17.29 default-ttl

- Integer
- Default: 3600

TTL to use when none is provided.

17.30 default-zsk-algorithm

- String
- Default: (empty)

Changed in version 4.1.0: Renamed from *default-zsk-algorithms*. Does no longer support multiple algorithm names.

The algorithm that should be used for the ZSK when running *pdnsutil secure-zone* or using the *Zone API endpoint* to enable DNSSEC. Must be one of:

- rsamd5
- dh
- dsa
- ecc
- rsasha1
- rsasha256
- rsasha512
- ecc-gost
- ecdsa256 (ECDSA P-256 with SHA256)
- ecdsa384 (ECDSA P-384 with SHA384)

- ed25519

Note: Actual supported algorithms depend on the crypto-libraries PowerDNS was compiled against. To check the supported DNSSEC algorithms in your build of PowerDNS, run `pdnsutil list-algorithms`.

17.31 default-zsk-size

- Integer
- Default: 0 (automatic default for *default-zsk-algorithm*)

The default keysize for the ZSK generated with *pdnsutil secure-zone*. Only relevant for algorithms with non-fixed keysizes (like RSA).

17.32 direct-dnskey

- Boolean
- Default: no

Read additional ZSKs from the records table/your BIND zonefile. If not set, DNSKEY records in the zonefiles are ignored.

17.33 disable-axfr

- Boolean
- Default: no

Do not allow zone transfers.

17.34 disable-axfr-rectify

- Boolean
- Default: no

Disable the rectify step during an outgoing AXFR. Only required for regression testing.

17.35 disable-syslog

- Boolean
- Default: no

Do not log to syslog, only to stdout. Use this setting when running inside a supervisor that handles logging (like systemd).

Warning: Do not use this setting in combination with *daemon* as all logging will disappear.

17.36 `disable-tcp`

- Boolean
- Default: no

Do not listen to TCP queries. Breaks RFC compliance.

17.37 `distributor-threads`

- Integer
- Default: 3

Number of Distributor (backend) threads to start per receiver thread. See *Performance and Tuning*.

17.38 `dnsname-processing`

- Boolean
- Default: no

Synthesise CNAME records from DNAME records as required. This approximately doubles query load. **Do not combine with DNSSEC!**

17.39 `dnssec-key-cache-ttl`

- Integer
- Default: 30

Seconds to cache DNSSEC keys from the database. A value of 0 disables caching.

17.40 `dnsupdate`

- Boolean
- Default: no

Enable/Disable DNS update (RFC2136) support. See *Dynamic DNS Update (RFC2136)* for more.

17.41 `do-ipv6-additional-processing`

- Boolean
- Default: yes

Perform AAAA additional processing. This sends AAAA records in the ADDITIONAL section when sending a referral.

17.42 domain-metadata-cache-ttl

- Integer
- Default: 60

Seconds to cache domain metadata from the database. A value of 0 disables caching.

17.43 edns-subnet-processing

- Boolean
- Default: no

Enables EDNS subnet processing, for backends that support it.

17.44 entropy-source

- Path
- Default: /dev/urandom

Entropy source file to use.

17.45 expand-alias

- Boolean
- Default: no
- Since: 4.1.0

If this is enabled, ALIAS records are expanded (synthesised to their A/AAAA).

If this is disabled (the default), ALIAS records will not expanded and the server will return NODATA for A/AAAA queries for such names.

note: *resolver* must also be set for ALIAS expansion to work!

note: In PowerDNS Authoritative Server 4.0.x, this setting did not exist and ALIAS was always expanded.

17.46 forward-dnsupdate

- Boolean
- Default: no

Forward DNS updates sent to a slave to the master.

17.47 forward-notify

- IP addresses, separated by commas

IP addresses to forward received notifications to regardless of master or slave settings.

Note: The intended use is in anycast environments where it might be necessary for a proxy server to perform the AXFR. The usual checks are performed before any received notification is forwarded.

17.48 guardian

- Boolean
- Default: no

Run within a guardian process. See *Guardian*.

17.49 include-dir

- Path

Directory to scan for additional config files. All files that end with .conf are loaded in order using POSIX as locale.

17.50 launch

- Backend names, separated by commas

Which backends to launch and order to query them in. Launches backends. In its most simple form, supply all backends that need to be launched. e.g.

```
launch=bind,gmysql,remote
```

If you find that you need to query a backend multiple times with different configuration, you can specify a name for later instantiations. e.g.:

```
launch=gmysql,gmysql:server2
```

In this case, there are 2 instances of the gmysql backend, one by the normal name and the second one is called 'server2'. The backend configuration item names change: e.g. gmysql-host is available to configure the host setting of the first or main instance, and gmysql-server2-host for the second one.

17.51 load-modules

- Paths, separated by commas

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 distributions.

17.52 local-address

- IPv4 Addresses, separated by commas or whitespace
- Default: 0.0.0.0

Local IP address to which we bind. 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.

17.53 log-timestamp

New in version 4.1.0.

- Bool
- Default: yes

When printing log lines to stdout, prefix them with timestamps. Disable this if the process supervisor timestamps these lines already.

Note: The systemd unit file supplied with the source code already disables timestamp printing

17.54 non-local-bind

- Boolean
- Default: no

Bind to addresses even if one or more of the *local-address*’s do not exist on this server. Setting this option will enable the needed socket options to allow binding to non-local addresses. This feature is intended to facilitate ip-failover setups, but it may also mask configuration issues and for this reason it is disabled by default.

17.55 lua-axfr-script

- String
- Default: empty

New in version 4.1.0.

Script to be used to edit incoming AXFRs, see *Modifying a slave zone using a script*

17.56 local-address-nonexist-fail

- Boolean
- Default: no

Fail to start if one or more of the *local-address*’s do not exist on this server.

17.57 local-ipv6

- IPv6 Addresses, separated by commas or whitespace
- Default: ‘::’

Local IPv6 address to which we bind. 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.

17.58 `local-ipv6-nonexist-fail`

- Boolean
- Default: no

Fail to start if one or more of the *local-ipv6* addresses do not exist on this server.

17.59 `local-port`

- Integer
- Default: 53

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

17.60 `log-dns-details`

- Boolean
- Default: no

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

17.61 `logging-facility`

If set to a digit, logging is performed under this LOCAL facility. See *Logging to syslog*. Do not pass names like 'local0'!

17.62 `loglevel`

- Integer
- Default: 4

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

17.63 `log-dns-queries`

- Boolean
- Default: no

Tell PowerDNS to log all incoming DNS queries. This will lead to a lot of logging! Only enable for debugging! Set *loglevel* to at least 5 to see the logs.

17.64 `lua-prequery-script`

- Path

Lua script to run before answering a query. This is a feature used internally for regression testing. The API of this functionality is not guaranteed to be stable, and is in fact likely to change.

17.65 master

- Boolean
- Default: no

Turn on master support. See *Master operation*.

17.66 max-cache-entries

- Integer
- Default: 1000000

Maximum number of entries in the query cache. 1 million (the default) will generally suffice for most installations. Starting with 4.1, the packet and query caches are distinct so you might also want to see `max-packet-cache-entries`.

17.67 max-ent-entries

- Integer
- Default: 100000

Maximum number of empty non-terminals to add to a zone. This is a protection measure to avoid database explosion due to long names.

17.68 max-nsec3-iterations

- Integer
- Default: 500

Limit the number of NSEC3 hash iterations

17.69 max-packet-cache-entries

- Integer
- Default: 1000000

Maximum number of entries in the packet cache. 1 million (the default) will generally suffice for most installations. This setting has been introduced in 4.1, previous used the `max-cache-entries` setting for both the packet and query caches.

17.70 max-queue-length

- Integer
- Default: 5000

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

17.71 max-signature-cache-entries

- Integer
- Default: 2^64 (on 64-bit systems)

Maximum number of signatures cache entries

17.72 max-tcp-connection-duration

- Integer
- Default: 0

Maximum time in seconds that a TCP DNS connection is allowed to stay open. 0 means unlimited. Note that exchanges related to an AXFR or IXFR are not affected by this setting.

17.73 max-tcp-connections

- Integer
- Default: 20

Allow this many incoming TCP DNS connections simultaneously.

17.74 max-tcp-connections-per-client

- Integer
- Default: 0

Maximum number of simultaneous TCP connections per client. 0 means unlimited.

17.75 max-tcp-transactions-per-conn

- Integer
- Default: 0

Allow this many DNS queries in a single TCP transaction. 0 means unlimited. Note that exchanges related to an AXFR or IXFR are not affected by this setting.

17.76 module-dir

- Path

Directory for modules. Default depends on PKGLIBDIR during compile-time.

17.77 negquery-cache-ttl

- Integer
- Default: 60

Seconds to store queries with no answer in the Query Cache. See ref:*query-cache*.

17.78 no-config

- Boolean
- Default: no

Do not attempt to read the configuration file.

17.79 no-shuffle

- Boolean
- Default: no

Do not attempt to shuffle query results, used for regression testing.

17.80 overload-queue-length

- Integer
- Default: 0 (disabled)

If this many packets are waiting for database attention, answer any new questions strictly from the packet cache.

17.81 reuseport

- Boolean
- Default: No

On Linux 3.9 and some BSD kernels the `SO_REUSEPORT` option allows each receiver-thread to open a new socket on the same port which allows for much higher performance on multi-core boxes. Setting this option will enable use of `SO_REUSEPORT` when available and seamlessly fall back to a single socket when it is not available. A side-effect is that you can start multiple servers on the same IP/port combination which may or may not be a good idea. You could use this to enable transparent restarts, but it may also mask configuration issues and for this reason it is disabled by default.

17.82 security-poll-suffix

- String
- Default: `secpoll.powerdns.com`.

Domain name from which to query security update notifications. Setting this to an empty string disables secpoll.

17.83 server-id

- String
- Default: The hostname of the server

This is the server ID that will be returned on an EDNS NSID query.

17.84 `only-notify`

- IP Ranges, separated by commas or whitespace
- Default: 0.0.0.0/0, ::/0

For type=MASTER zones (or SLAVE zones with slave-notify enabled) PowerDNS automatically sends NOTIFYs to the name servers specified in the NS records. By specifying networks/mask as whitelist, the targets can be limited. The default is to notify the world. To completely disable these NOTIFYs set `only-notify` to an empty value. Independent of this setting, the IP addresses or netmasks configured with *also-notify* and ALSO-NOTIFY domain metadata always receive AXFR NOTIFYs.

Note: Even if NOTIFYs are limited by a netmask, PowerDNS first has to resolve all the hostnames to check their IP addresses against the specified whitelist. The resolving may take considerable time, especially if those hostnames are slow to resolve. If you do not need to NOTIFY the slaves defined in the NS records (e.g. you are using another method to distribute the zone data to the slaves), then set *only-notify* to an empty value and specify the notification targets explicitly using *also-notify* and/or *ALSO-NOTIFY* domain metadata to avoid this potential bottleneck.

17.85 `out-of-zone-additional-processing`

- Boolean
- Default: yes

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.

The docs had previously indicated that the default was “no”, but the default has been “yes” since 2005.

17.86 `outgoing-axfr-expand-alias`

- Boolean
- Default: no

If this is enabled, ALIAS records are expanded (synthesised to their A/AAAA) during outgoing AXFR. This means slaves will not automatically follow changes in those A/AAAA records unless you AXFR regularly!

If this is disabled (the default), ALIAS records are sent verbatim during outgoing AXFR. Note that if your slaves do not support ALIAS, they will return NODATA for A/AAAA queries for such names.

17.87 `prevent-self-notification`

- Boolean
- Default: yes

PowerDNS Authoritative Server attempts to not send out notifications to itself in master mode. In very complicated situations we could guess wrong and not notify a server that should be notified. In that case, set prevent-self-notification to “no”.

17.88 `query-cache-ttl`

- Integer
- Default: 20

Seconds to store queries with an answer in the Query Cache. See [Query Cache](#).

17.89 `query-local-address`

- IPv4 Address
- Default: 0.0.0.0

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

17.90 `query-local-address6`

- IPv6 Address
- Default: '::'

Source IP address for sending IPv6 queries.

17.91 `query-logging`

- Boolean
- Default: no

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

17.92 `queue-limit`

- Integer
- Default: 1500

Maximum number of milliseconds to queue a query. See [Performance and Tuning](#).

17.93 `receiver-threads`

- Integer
- Default: 1

Number of receiver (listening) threads to start. See [Performance and Tuning](#).

17.94 recursive-cache-ttl

- Integer
- Default: 10
- Removed in: 4.1.0

Seconds to store recursive packets in the *Packet Cache*.

17.95 recursor

- IP Address

Deprecated since version 4.1.0.

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

17.96 resolver

- IP Addresses with optional port, separated by commas
- Added in: 4.1.0

Use these resolver addresses for ALIAS and the internal stub resolver. If this is not set, `/etc/resolv.conf` is parsed for upstream resolvers.

17.97 retrieval-threads

- Integer
- Default: 2

Number of AXFR slave threads to start.

17.98 setgid

- String

If set, change group id to this gid for more security. See *Security of PowerDNS*.

17.99 setuid

- String

If set, change user id to this uid for more security. See *Security of PowerDNS*.

17.100 slave

- Boolean
- Default: no

Turn on slave support. See *Slave operation*.

17.101 `slave-cycle-interval`

- Integer
- 60

On a master, this is the amounts of seconds between the master checking the SOA serials in its database to determine to send out NOTIFYs to the slaves. On slaves, this is the number of seconds between the slave checking for updates to zones.

17.102 `slave-renotify`

- Boolean
- Default: no

This setting will make PowerDNS renotify the slaves after an AXFR is *received* from a master. This is useful when using when running a signing-slave.

17.103 `signing-threads`

- Integer
- Default: 3

Tell PowerDNS how many threads to use for signing. It might help improve signing speed by changing this number.

17.104 `soa-expire-default`

- Integer
- Default: 604800

Default *SOA* expire.

17.105 `soa-minimum-ttl`

- Integer
- Default: 3600

Default *SOA* minimum ttl.

17.106 `soa-refresh-default`

- Integer
- Default: 10800

Default *SOA* refresh.

17.107 `soa-retry-default`

- Integer
- Default: 3600

Default *SOA* retry.

17.108 `socket-dir`

- Path

Where the controlsocket will live. The default depends on `LOCALSTATEDIR` during compile-time (usually `/var/run` or `/run`). See *Control Socket*.

This path will also contain the pidfile for this instance of PowerDNS called `pdns.pid` by default. See *config-name* and *Virtual Hosting* how this can differ.

17.109 `tcp-control-address`

- IP Address

Address to bind to for TCP control.

17.110 `tcp-control-port`

- Integer
- Default: 53000

Port to bind to for TCP control.

17.111 `tcp-control-range`

- IP Ranges, separated by commas or whitespace

Limit TCP control to a specific client range.

17.112 `tcp-control-secret`

- String

Password for TCP control.

17.113 `tcp-fast-open`

- Integer
- Default: 0 (Disabled)

New in version 4.1.0.

Enable TCP Fast Open support, if available, on the listening sockets. The numerical value supplied is used as the queue size, 0 meaning disabled.

17.114 `tcp-idle-timeout`

- Integer
- Default: 5

Maximum time in seconds that a TCP DNS connection is allowed to stay open while being idle, meaning without PowerDNS receiving or sending even a single byte.

17.115 `traceback-handler`

- Boolean
- Default: yes

Enable the Linux-only traceback handler.

17.116 `trusted-notification-proxy`

- String

IP address of incoming notification proxy

17.117 `udp-truncation-threshold`

- Integer
- Default: 1680

EDNS0 allows for large UDP response datagrams, which can potentially raise performance. Large responses however also have downsides in terms of reflection attacks. Up till PowerDNS Authoritative Server 3.3, the truncation limit was set at 1680 bytes, regardless of EDNS0 buffer size indications from the client. Beyond 3.3, this setting makes our truncation limit configurable. Maximum value is 65535, but values above 4096 should probably not be attempted.

17.118 `version-string`

- Any of: `anonymous`, `powerdns`, `full`, String
- Default: full

When queried for its version over DNS (`dig chaos txt version.bind @pdns.ip.address`), 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.

17.119 `webserver`

- Boolean
- Default: no

Start a webserver for monitoring. See *Performance and Tuning*".

Changed in version 4.1.0: It was necessary to enable the webserver to use the REST API, this is no longer the case.

17.120 `webserver-address`

- IP Address
- Default: 127.0.0.1

IP Address for webserver/API to listen on.

17.121 `webserver-allow-from`

- IP ranges, separated by commas or whitespace
- Default: 127.0.0.1,::1

Changed in version 4.1.0: Default is now 127.0.0.1,::1, was 0.0.0.0,::/0 before.

Webserver/API access is only allowed from these subnets.

17.122 `webserver-password`

- String

The plaintext password required for accessing the webserver.

17.123 `webserver-port`

- Integer
- Default: 8001

The port where webserver/API will listen on.

17.124 `webserver-print-arguments`

- Boolean
- Default: no

If the webserver should print arguments.

17.125 `write-pid`

- Boolean
- Default: yes

If a PID file should be written.

17.126 `xfr-max-received-mbytes`

- Integer
- Default: 100

Specifies the maximum number of received megabytes allowed on an incoming AXFR/IXFR update, to prevent resource exhaustion. A value of 0 means no restriction.

SECURITY ADVISORIES

All security advisories for the PowerDNS Authoritative Server are listed here.

18.1 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 ('pdns_recursor') 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.

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.

18.2 PowerDNS Security Advisory 2008-03: Some PowerDNS Configurations can be forced to restart remotely

- 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.

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 “*Security*” 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.

18.3 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.

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.

18.4 PowerDNS Security Advisory 2015-01: Label decompression bug can cause crashes or CPU spikes

- CVE: CVE-2015-1868 (original), CVE-2015-5470 (update)
- Date: 23rd of April 2015, updated 7th of July 2015
- Credit: Aki Tuomi, Toshifumi Sakaguchi
- Affects: PowerDNS Recursor versions 3.5 and up; Authoritative Server 3.2 and up
- Not affected: Recursor 3.6.4; Recursor 3.7.3; Auth 3.3.3; Auth 3.4.5
- Severity: High
- Impact: Degraded service
- Exploit: This problem can be triggered by sending queries for specifically configured domains, or by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to any of the non-affected versions
- Workaround: Run your Recursor under a supervisor. Exposure can be limited by configuring the `allow-from` in `./recursor/settings.md#allow-from` setting so only trusted users can query your nameserver. There is no workaround for the Authoritative server.

A bug was discovered in our label decompression code, making it possible for names to refer to themselves, thus causing a loop during decompression. On some platforms, this bug can be abused to cause crashes. On all platforms, this bug can be abused to cause service-affecting CPU spikes.

We recommend that all users upgrade to a corrected version if at all possible. Alternatively, if you want to apply a minimal fix to your own tree, please [find patches here](#).

As for workarounds, for the Recursor: only clients in `allow-from` are able to trigger the degraded service, so this should be limited to your userbase; further, we recommend running your critical services under supervision such as `systemd`, `supervisord`, `daemontools`, etc.

There is no workaround for the Authoritative Server.

We want to thank Aki Tuomi for noticing this in production, and then digging until he got to the absolute bottom of what at the time appeared to be a random and spurious failure.

We want to thank Toshifumi Sakaguchi for further investigation into the issue after the initial announcement, and for demonstrating to us quite clearly the CPU spike issues.

Update 7th of July 2015: Toshifumi Sakaguchi discovered that the original fix was insufficient in some cases. Updated versions of the Authoritative Server and Recursor [were released](#) on the 9th of June. Minimal patches are [available](#). The insufficient fix was assigned CVE-2015-5470.

18.5 PowerDNS Security Advisory 2015-02: Packet parsing bug can cause thread or process abortion

- CVE: CVE-2015-5230
- Date: 2nd of September 2015
- Credit: Pyry Hakulinen and Ashish Shukla at Automattic
- Affects: PowerDNS Authoritative Server 3.4.0 through 3.4.5
- Not affected: PowerDNS Authoritative Server 3.4.6
- Severity: High
- Impact: Degraded service or Denial of service
- Exploit: This problem can be triggered by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Run the Authoritative Server inside a supervisor when `distributor-threads` is set to 1 to prevent Denial of Service. No workaround for the degraded service exists

A bug was found in our DNS packet parsing/generation code, which, when exploited, can cause individual threads (disabling service) or whole processes (allowing a supervisor to restart them) to crash with just one or a few query packets.

PowerDNS Authoritative Server 3.4.0-3.4.5 are affected. No other versions are affected. The PowerDNS Recursor is not affected.

PowerDNS Authoritative Server 3.4.6 contains a fix to this issue. A minimal patch is [available here](#).

This issue is entirely unrelated to [Security Advisory 2015-01/CVE-2015-1868](#).

We'd like to thank Pyry Hakulinen and Ashish Shukla at Automattic for finding and subsequently reporting this bug.

18.6 PowerDNS Security Advisory 2015-03: Packet parsing bug can lead to crashes

- CVE: CVE-2015-5311
- Date: November 9th 2015
- Credit: Christian Hofstaedtler of Deduktiva GmbH
- Affects: PowerDNS Authoritative Server 3.4.4 through 3.4.6
- Not affected: PowerDNS Authoritative Server 3.3.x and 3.4.7 and up
- Severity: High
- Impact: Degraded service or Denial of service
- Exploit: This problem can be triggered by sending specially crafted query packets

- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: run the process inside the guardian or inside a supervisor

A bug was found using `afl-fuzz` in our packet parsing code. This bug, when exploited, causes an assertion error and consequent termination of the `pdns_server` process, causing a Denial of Service.

When the PowerDNS Authoritative Server is run inside the guardian (`--guardian`), or inside a supervisor like `supervisord` or `systemd`, it will be automatically restarted, limiting the impact to a somewhat degraded service.

PowerDNS Authoritative Server 3.4.4 - 3.4.6 are affected. No other versions are affected. The PowerDNS Recursor is not affected.

PowerDNS Authoritative Server 3.4.7 contains a fix to this issue. A minimal patch is [available here](#).

This issue is unrelated to the issues in our previous two Security Announcements (2015-01 and 2015-02).

We'd like to thank Christian Hofstaedtler of Deduktiva GmbH for finding and reporting this issue.

18.7 PowerDNS Security Advisory 2016-01: Crafted queries can cause unexpected backend load

- CVE: CVE-2016-5426, CVE-2016-5427
- Date: 9th of September 2016
- Credit: Florian Heinz and Martin Kluge
- Affects: PowerDNS Authoritative Server up to and including 3.4.9
- Not affected: PowerDNS Authoritative Server 3.4.10, 4.x
- Severity: Medium
- Impact: Degraded service or Denial of service
- Exploit: This problem can be triggered by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Run `dnsdist` with the rules provided below in front of potentially affected servers, or dimension the backend capacity so that it can handle the increased load.

Two issues have been found in PowerDNS Authoritative Server allowing a remote, unauthenticated attacker to cause an abnormal load on the PowerDNS backend by sending crafted DNS queries, which might result in a partial denial of service if the backend becomes overloaded. SQL backends for example are particularly vulnerable to this kind of unexpected load if they have not been dimensioned for it. The first issue is based on the fact that PowerDNS Authoritative Server accepts queries with a qname's length larger than 255 bytes. This issue has been assigned CVE-2016-5426. The second issue is based on the fact that PowerDNS Authoritative Server does not properly handle dot inside labels. This issue has been assigned CVE-2016-5427. Both issues have been addressed by this [commit](#).

PowerDNS Authoritative Server up to and including 3.4.9 is affected. No other versions are affected. The PowerDNS Recursor is not affected.

`dnsdist` can be used to block crafted queries, using `QNameWireLengthRule()` to block queries with a qname larger than 255 bytes and `QNameLabelsCountRule()` to block queries with a very large amount of labels. Please note that restricting the number of labels in a query might lead to unexpected issues, especially with DNSSEC-enabled domains.

We'd like to thank Florian Heinz and Martin Kluge for finding and subsequently reporting this issue.

18.8 PowerDNS Security Advisory 2016-02: Crafted queries can cause abnormal CPU usage

- CVE: CVE-2016-7068
- Date: December 15th 2016
- Credit: Florian Heinz and Martin Kluge
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1, PowerDNS Recursor up to and including 3.7.3, 4.0.3
- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2 and PowerDNS Recursor 3.7.4, 4.0.4
- Severity: Medium
- Impact: Degraded service or Denial of service
- Exploit: This issue can be triggered by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Run `dnscat` with the rules provided below in front of potentially affected servers.

An issue has been found in PowerDNS allowing a remote, unauthenticated attacker to cause an abnormal CPU usage load on the PowerDNS server by sending crafted DNS queries, which might result in a partial denial of service if the system becomes overloaded. This issue is based on the fact that the PowerDNS server parses all records present in a query regardless of whether they are needed or even legitimate. A specially crafted query containing a large number of records can be used to take advantage of that behaviour. This issue has been assigned CVE-2016-7068.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. PowerDNS Recursor up to and including 3.7.3 and 4.0.3 are affected.

`dnscat` can be used to block crafted queries, using `RecordsCountRule()` and `RecordsTypeCountRule()` to block queries with crafted records.

For those unable to upgrade to a new version, a minimal patch is [available](#)

We would like to thank Florian Heinz and Martin Kluge for finding and subsequently reporting this issue.

18.9 PowerDNS Security Advisory 2016-03: Denial of service via the web server

- CVE: CVE-2016-7072
- Date: December 15th 2016
- Credit: Mongo
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1
- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2
- Severity: Medium
- Impact: Degraded service or Denial of service
- Exploit: This issue can be triggered by opening a large number of simultaneous connections to the web server
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

- Workaround: Disable the web server, or restrict access to it via a firewall.

An issue has been found in PowerDNS Authoritative Server allowing a remote, unauthenticated attacker to cause a denial of service by opening a large number of TCP connections to the web server. If the web server runs out of file descriptors, it triggers an exception and terminates the whole PowerDNS process. While it's more complicated for an unauthorized attacker to make the web server run out of file descriptors since its connection will be closed just after being accepted, it might still be possible. This issue has been assigned CVE-2016-7072.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. The PowerDNS Recursor is not affected.

For those unable to upgrade to a new version, a minimal patch is [available](#)

We would like to thank Mongo for finding and subsequently reporting this issue.

18.10 PowerDNS Security Advisory 2016-04: Insufficient validation of TSIG signatures

- CVE: CVE-2016-7073 CVE-2016-7074
- Date: December 15th 2016
- Credit: Mongo
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1, PowerDNS Recursor from 4.0.0 and up to and including 4.0.3
- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2, PowerDNS Recursor < 4.0.0, 4.0.4
- Severity: Medium
- Impact: Zone content alteration
- Exploit: This problem can be triggered by an attacker in position of man-in-the-middle
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

Two issues have been found in PowerDNS Authoritative Server allowing an attacker in position of man-in-the-middle to alter the content of an AXFR because of insufficient validation of TSIG signatures. The first issue is a missing check of the TSIG time and fudge values in `AXFRRetriever`, leading to a possible replay attack. This issue has been assigned CVE-2016-7073. The second issue is a missing check that the TSIG record is the last one, leading to the possibility of parsing records that are not covered by the TSIG signature. This issue has been assigned CVE-2016-7074.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. PowerDNS Recursor from 4.0.0 up to and including 4.0.3 are affected.

For those unable to upgrade to a new version, a minimal patch is [available](#)

We would like to thank Mongo for finding and subsequently reporting this issue.

18.11 PowerDNS Security Advisory 2016-05: Crafted zone record can cause a denial of service

- CVE: CVE-2016-2120
- Date: December 15th 2016
- Credit: Mathieu Lafon
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1

- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2
- Severity: Medium
- Impact: Denial of service
- Exploit: This issue can be triggered by inserting a specially crafted record in a zone
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in PowerDNS Authoritative Server allowing an authorized user to crash the server by inserting a specially crafted record in a zone under their control then sending a DNS query for that record. The issue is due to an integer overflow when checking if the content of the record matches the expected size, allowing an attacker to cause a read past the buffer boundary. This issue has been assigned CVE-2016-2120.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. The PowerDNS Recursor is not affected.

For those unable to upgrade to a new version, a minimal patch is [available](#)

We would like to thank Mathieu Lafon for finding and subsequently reporting this issue.

18.12 PowerDNS Security Advisory 2017-04: Missing check on API operations

- CVE: CVE-2017-15091
- Date: November 27th 2017
- Credit: everyman
- Affects: PowerDNS Authoritative up to and including 4.0.4, 3.4.11
- Not affected: PowerDNS Authoritative 4.0.5
- Severity: Low
- Impact: Denial of service
- Exploit: This problem can be triggered by an attacker with valid API credentials
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in the API component of PowerDNS Authoritative, where some operations that have an impact on the state of the server are still allowed even though the API has been configured as read-only via the [api-readonly](#) keyword. This missing check allows an attacker with valid API credentials to flush the cache, trigger a zone transfer or send a NOTIFY. This issue has been assigned CVE-2017-15091.

PowerDNS Authoritative up to and including 4.0.4 and 3.4.11 are affected.

For those unable to upgrade to a new version, a minimal patch is [available](#)

We would like to thank everyman for finding and subsequently reporting this issue.

18.13 Older security advisories

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.

CHANGELOGS

The changelogs for the PowerDNS Authoritative Servr are split between release trains.

19.1 Changelogs for 4.1.x

19.1.1 4.1.1

Released: 16th of February 2018

This is the second release in the 4.1 train.

This is a bug-fix only release, with fixes to the LDAP and MySQL backends, the `pdnsutil` tool, and PDNS internals.

Changes since 4.1.1:

Bug Fixes

- Backport: forbid label compression in alias wire format ¶ References: #6028, pull request 6260
- Include `unistd.h` for `chroot(2)` et al. (Florian Obser) ¶ References: pull request 6077
- Auth: fix out of bounds exception in `caa` processing, fixes #6089 ¶ References: pull request 6103
- Add the missing `<sys/time.h>` include to `mplexer.hh` for `struct timeval` ¶ References: #6040, pull request 6041
- Auth: init `openssl` and `libsodium` before `chrooting` in `pdnsutil` ¶ References: #6125, pull request 6129
- Auth: always bind the results array after executing a `mysql` statement ¶ References: #6115, pull request 6134
- Ldap: fix `getdomaininfo()` to set `this` as `di.backend` (Grégory Oestreicher) ¶ References: pull request 6048
- Ldapbackend: fix listing zones incl. `axfr` (Chris Hofstaedtler) ¶ References: #6060, #6097, pull request 6122
- Ixfr: correct behavior of dealing with dns name with multiple records (Leon Xu) ¶ References: pull request 6172

19.1.2 4.1.0

Released: 30th of November 2017

This is the first release in the 4.1 train.

The full release notes can be read [on the blog](#).

The 4.1 release is a major upgrade for the Authoritative Server featuring many improvements and speedups:

- Improved performance: 400% speedup in some scenarios,
- Crypto API: DNSSEC fully configurable via RESTful API,
- Improved documentation,
- Database related improvements,
- Enhanced tooling,
- Support for TCP Fast Open,
- Support for non-local bind,
- Support for Botan 2.x (and removal of support for Botan 1.10),
- Our packages now ship with PKCS #11 support.

Recursor passthrough removal: This will impact many installations, and we realize this may be painful, but it is necessary. Previously, the PowerDNS Authoritative Server contained a facility for sending recursion desired queries to a resolving backend, possibly after first consulting its local cache. This feature ('recursor=') was frequently confusing and also delivered inconsistent results, for example when a query ended up referring to a CNAME that was outside of the Authoritative Server's knowledge. To read more about this please see the blog post mentioned above or read the [migration guide](#).

Changes since 4.1.0-rc3:

Removed Features

- Remove deprecated SOA-EDIT values: INCEPTION and INCEPTION-WEEK. ¶ References: [pull request 6004](#)

Improvements

- Make the /cryptokeys endpoint consistently use CryptoKey objects. ¶ References: [#5862](#), [pull request 5964](#)
- Report remote IP when SOA query comes back with empty question section. ¶ References: [#5974](#), [pull request 5976](#)

Bug Fixes

- Deny cache flush, zone retrieve and notify if the API is read-only. ¶ References: [pull request 6007](#)
- Fix hang when PATCHing zone during rectify. ¶ References: [pull request 5968](#)

19.1.3 4.1.0-rc3

Released: 17th of November 2017

This is the third release candidate of the PowerDNS Authoritative Server in the 4.1 release train.

This release features various bug fixes and some improvements to `pdnsutil`.

New Features

- Make it possible to disable DNSSEC via the API, this is equivalent to doing `pdnsutil disable-dnssec`. ¶ References: [#5909](#), [#5910](#), [pull request 5936](#)
- Add `add-meta` command to `pdnsutil` that can be used to append to existing metadata without clobbering it. ¶ References: [#5853](#), [pull request 5883](#)

Improvements

- Better support for deleting entries in NetmaskTree and NetmaskGroup. ¶ References: [pull request 5616](#)
- Throw exception for metadata endpoint with wrong zone. Before, We would happily accept this POST. ¶ References: [pull request 5935](#)
- Warn if records in a zone are occluded. ¶ References: [#5948](#), [#5949](#), [#3059](#), [pull request 5879](#)

Bug Fixes

- Use `_exit()` when we really really want to exit, for example after a fatal error. This stops us dying while we die. A call to `exit()` will trigger destructors, which may paradoxically stop the process from exiting, taking down only one thread, but harming the rest of the process. ¶ References: [pull request 5917](#)
- Fix messages created by `pdnsutil generate-tsig-key`. ¶ References: [#5849](#), [pull request 5884](#)
- Add back missing output details to `rectifyZone`. ¶ References: [#5903](#), [pull request 5928](#)
- Use 302 redirects in the webserver for ringbuffer reset or resize. With the current 301 redirect it is only possible to reset or resize once. Every next duplicate action is replaced by the destination cached in the browser. ¶ References: [pull request 5905](#)

19.1.4 4.1.0-rc2

Released: 3rd of November 2017

This is the second release candidate of the PowerDNS Authoritative Server in the 4.1 release train.

This release has several performance improvements, stability and correctness fixes.

New Features

- Rectify zones via the API. (Nils Wisiol)
 - Move the `pdnsutil` rectification code to the `DNSSECKeeper`
 - Generate DNSSEC keys for a zone when “`dnssec`” is true in an API POST/PATCH for zones
 - Rectify DNSSEC zones after POST/PATCH when API-RECTIFY metadata is 1
 - Allow setting this metadata via the “`api-rectify`” param in a Zone object
 - Show “`nsec3param`” and “`nsec3narrow`” in Zone API responses
 - Add an “`rrsets`” request parameter for a zone to skip sending RRSets in the response
 - Add rectify endpoint in the API

¶ References: [#5712](#), [#3417](#), [pull request 5779](#)

- Add *PKCS#11* support to packages on Operating Systems that support it. ¶ References: [pull request 5665](#)

Improvements

- Add support for Botan 2.x and drop support for Botan 1.10 (the latter thanks to Kees Monshouwer). ¶ References: [#5889](#), [#2250](#), [#5797](#), [#5734](#), [pull request 5498](#)
- Fix issues when b2b-migrating from the BIND backend to a database:
 - No masters were set in the target db ([#5807](#))
 - Only the last master in the list of masters would be added to the target database
 - The BIND backend was not fully aware of native zones

🔗 References: [#5115](#), [#5807](#), [pull request 5810](#)

- Add support for new record types to the LDAP backend. 🔗 References: [pull request 5584](#)
- Add *log-timestamp* option. This option can be used to disable printing timestamps to stdout, this is useful when using systemd-journald or another supervisor that timestamps stdout by itself. As the logs will not have 2 timestamps. 🔗 References: [pull request 5842](#)
- Stop doing individual RRSIG queries during outbound AXFR. (Kees Monshouwer) 🔗 References: [#5767](#), [pull request 5838](#)

Bug Fixes

- Improve trailing dot handling internally which lead to a segfault in pdnsutil before. 🔗 References: [#5673](#), [pull request 5684](#)
- Treat requestor's payload size lower than 512 as equal to 512. Before, we did not follow [RFC 6891 section 6.2.3](#) correctly. 🔗 References: [pull request 5678](#)
- Correctly purge entries from the caches after a transfer. Since the QC/PC split up, we only removed entries for the AXFR'd domain from the packet cache, not the query cache. We also did not remove entries in case of IXFR. 🔗 References: [#5767](#), [pull request 5766](#)
- When throwing because of bogus content in the tinydns database, report the offending name+type so the admin can find the offending record. 🔗 References: [pull request 5791](#)
- For zone PATCH requests, add new X-PDNS-Old-Serial and X-PDNS-New-Serial response headers with the zone serials before and after the changes. 🔗 References: [pull request 5696](#)
- Make default options singular and use defaults in Cryptokey API-endpoint 🔗 References: [pull request 5704](#)
- Remove printing of DS records from `pdnsutil export-zone-dnskey`. This was not only inconsistent behaviour but also done incorrectly. 🔗 References: [#5719](#), [pull request 5729](#)
- Make bindbackend startTransaction to return false when it has failed. (Aki Tuomi) 🔗 References: [pull request 5702](#)
- Log the needed size when a MySQL result was truncated. 🔗 References: [#5675](#), [pull request 5820](#)
- Remove "" around secpoll result which fixes `pdns_control show security-status` not working. 🔗 References: [#5692](#), [pull request 5710](#)
- Make the auth also publish CDS/CDNSKEY records for inactive keys, as this is needed to roll without double sigs. 🔗 References: [#5721](#), [pull request 5722](#)
- Fix a crash when getting a public GOST key if the private one is not set. 🔗 References: [pull request 5734](#)
- Ignore SOA-EDIT for PRESIGNED zones. 🔗 References: [pull request 5815](#)

19.1.5 4.1.0-rc1

Released: 31st of August 2017

This is the first release candidate of the PowerDNS Authoritative Server in the 4.1 release train.

New Features

- Add TCP management options described in [section 10 of RFC 7766](#). 🔗 References: [pull request 4624](#)
- Add TCP Fast Open support. 🔗 References: [#5129](#), [pull request 5137](#)
- Hash the entire query in the packet cache, split caches. This makes the authoritative server pass the EDNS compliance test.

Add cache hit/miss statistics (Kees Monshouwer). ¶ References: [#4204](#), [pull request 5132](#), [pull request 5258](#)

- Add an adjustable statistics interval (@phonedph1). ¶ References: [#3781](#), [pull request 5190](#), [pull request 5271](#)
- Add option to set a global *lua-axfr-script* (Kees Monshouwer). ¶ References: [pull request 5316](#)
- Allow forwarding of NOTIFY messages using *forward-notify* (@DrRemorse). ¶ References: [pull request 4964](#), [pull request 1701](#), [pull request 4965](#)
- Add API endpoints for Domain metadata (Christian Kröger). ¶ References: [pull request 5038](#), [pull request 4093](#)
- Implement *CryptoKey* in the API (Wolfgang Studier, @MrM0nkey, Tudor Soroceanu, Benjamin Zengin). ¶ References: [#706](#), [pull request 4106](#)
- calidns: add `-increment` and `-want-recursion` flags. ¶ References: [pull request 5339](#)
- Allow the use of a *Lua script* to validate DNS Update requests (Aki Tuomi). ¶ References: [pull request 4058](#)
- Send a notification to all slave servers after every dnsupdate (Kees Monshouwer, Florian Obser). ¶ References: [#4821](#), [pull request 5263](#), [pull request 5264](#), [pull request 5321](#)
- Support “native” zones in the BIND backend. ¶ References: [#1284](#), [pull request 5115](#)
- Many improvements and additions to the LDAP backend (Grégory Oestreicher). ¶ References: [#3358](#), [pull request 4477](#)
- Support 2-character country codes and the MaxMind cities database in the GeoIP backend (Aki Tuomi). ¶ References: [#4122](#), [#5255](#), [pull request 5266](#), [pull request 5270](#), [pull request 5269](#)
- Add function to the MyDNS backend to allow backend-to-backend migrations (Aki Tuomi). ¶ References: [pull request 5043](#)
- Support the SMIMEA RRTYPE. ¶ References: [pull request 5379](#)

Removed Features

- Remove recursion. See *Migrating from using recursion on the Authoritative Server to using a Recursor* for migration strategies (Kees Monshouwer). ¶ References: [#2380](#), [#3337](#), [#4616](#), [#4238](#), [#2606](#), [#4315](#), [pull request 4752](#)
- Remove the experimental Lua Policy Engine (Aki Tuomi). ¶ References: [pull request 5468](#)

Improvements

- Revamp and clean label compression code. Speeds up large packet creation by ~40%. ¶ References: [pull request 4373](#)
- Apply *non-local-bind* to *query-local-address* and *query-local-address6* when possible. ¶ References: [#4299](#), [pull request 4332](#)
- A number of fixes and improvements that are difficult to untangle:
 - Remove the ASCII *DNSResourceRecord* from the hot path of packet assembly.
 - Hash the storage of records in the BindBackend.
 - Hash the packetcache.
 - Fix some bugs in the LDAP backend and in the MyDNS backend.
 - Make the randombackend go ‘native’ and directly supply records that can be sent to packets
 - The performance benefit of this PR is measured in “factors” for being a root-server.

🔗 References: [pull request 4467](#), [pull request 4492](#)

- Improve cleaning, remove an unnecessary lock and improve performance of the packetcache (Kees Monshouwer). 🔗 References: [#4503](#), [pull request 4504](#)
- Improve SOA records caching (Kees Monshouwer). 🔗 References: [pull request 4485](#)
- Make sure AXFR only deletes records from a SLAVE domain in a multi backend setup (Kees Monshouwer). 🔗 References: [pull request 4829](#)
- Tidy up UeberBackend (Christian Hofstaedtler). 🔗 References: [pull request 4908](#)
- Improve API performance by instantiating only one DNSSECKeeper per request. 🔗 References: [pull request 4944](#)
- Incremental backoff for failed slave checks.

When a SOA record for a slave domain can't be retrieved, use an increasing interval between checking the domain again. This prevents hammering down on already busy servers. 🔗 References: [#602](#), [#349](#), [pull request 4953](#)

- Remove d_place from DNSResourceRecord (Christian Hofstaedtler). 🔗 References: [pull request 4549](#)
- Add an option to allow AXFR of zones with a different (higher/lower) serial (Kees Monshouwer). 🔗 References: [pull request 5169](#)
- Use the *resolver* setting for the stub resolver, use resolv.conf as fallback. 🔗 References: [#4655](#), [pull request 5112](#)
- Re-implement the AXFR Filter with LuaContext (Aki Tuomi). 🔗 References: [pull request 5250](#)
- Allow control socket to listen on IPv6 (@Gibheer). 🔗 References: [pull request 5387](#)
- Fix typo in two log messages (Ruben Kerkhof). 🔗 References: [pull request 5523](#)
- Update YaHTTP (to fix a warning reported by Coverity). 🔗 References: [pull request 5542](#)
- Clarify how we check the return value of std::string::find() (reported by Coverity). 🔗 References: [pull request 5541](#)
- Wrap the webserver's and Resolver::tryGetSOASerial objects into smart pointers. 🔗 References: [pull request 5543](#)
- SSqL: Use unique_ptr for statements (Aki Tuomi). 🔗 References: [pull request 4692](#)
- Fix libatomic detection on ppc64 (Sander Hoentjen). 🔗 References: [pull request 5599](#)
- Switch the default webserver's ACL to "127.0.0.1, ::1". 🔗 References: [pull request 5588](#)
- NOTIMP is only appropriate for an unsupported opcode (Kees Monshouwer). 🔗 References: [pull request 5611](#)
- Catch DNSName exception in the Zoneparser. 🔗 References: [pull request 5641](#)
- Listen on 127.0.0.1 during regression tests (@tcely). 🔗 References: [pull request 5583](#)
- Enable the webserver when *api* is 'yes' (Christian Hofstaedtler). 🔗 References: [#4290](#), [pull request 4408](#)
- Prevent sending nameservers list and zone-level NS in rrsets in the API (Christian Hofstaedtler). 🔗 References: [#4132](#), [pull request 4751](#)
- Forbid mixing CNAMEs and other RRsets in the API (Christian Hofstaedtler). 🔗 References: [#5305](#), [pull request 5389](#)
- Prevent duplicate records in single RRset (Christian Hofstaedtler). 🔗 References: [pull request 4195](#)
- Implement subcommand printing all KSK DS records in pdnsutil (Jonas Wielicki). 🔗 References: [#4005](#), [pull request 4007](#)
- Allow setting the account of a zone via pdnsutil (Tuxis Internet Engineering).

🔗 References: [pull request 4584](#)

- Print “\$ORIGIN .” on `pdnsutil list-zone`, so the output can be used in `pdnsutil load-zone` (Tuxis Internet Engineering). [References: pull request 4719](#)
- `pdnsutil`: clarify error message when set-presigned fails with DNSSEC disabled (Peter Thomassen). [References: pull request 4478](#)
- `pdnsutil`: Validate names with address records to be valid hostnames (Håkan Lindqvist). [References: pull request 3913](#)
- Correct `pdnsutil` help output for `add-zone-key`. [References: pull request 5118](#)
- Check for valid hostnames in SRV, NS and MX records. [References: #512, pull request 5062](#)
- Disable ALIAS expansion by default. [References: #5119, pull request 5182](#)
- Make the zone parser adhere to [RFC 2308](#) with regards to implicit TTLs.
Existing zone files may now be interpreted differently. Specifically, where we previously used the SOA minimum field for the default TTL if none was set explicitly, or no \$TTL was set, we now use the TTL from the previous line. [References: pull request 5094](#)
- `mydnsbackend`: Initialize `d_query_stmt` (Aki Tuomi). [References: pull request 5605](#)
- Enable setting custom `pgsql` connection parameters, like TLS parameters (Tarjei Husøy). [References: pull request 4711](#)
- Use `pkg-config` to detect PostgreSQL libraries. [References: #5193, #2358, pull request 5121, pull request 5221](#)
- Use BIGSERIAL for `records.id` in the `gpgsql` backend (Arsen Stasic). [References: pull request 5426](#)
- Ship `ldapbackend` schema files in tarball (Christian Hofstaedtler). [References: pull request 5509](#)
- Add ability to have service record for apex record and any other static record (Aki Tuomi). [References: pull request 5548](#)
- Report query statistics as full numbers, not scientific notation in the webserver. [References: #1844, pull request 5116](#)
- Schema changes for MySQL / MariaDB and PostgreSQL to for storage requirements of various versions (Kees Monshouwer). [References: pull request 5518](#)

Bug Fixes

- Fix compilation on systems with Boost < 1.54 [References: pull request 4424](#)
- Fix possible variable shadowing (Kees Monshouwer, Christian Hofstaedtler). [References: #4546, pull request 4548, pull request 4560](#)
- Fix `getaddrinfo()` returning address in triplicate. [References: pull request 4855](#)
- Turn exception in a `qthread` into an error instead of a crash. [References: pull request 5117](#)
- Remove duplicate `dns2_tolower()` function and move `ascii`-related function to one file (Thiago Farina). [References: pull request 5249, pull request 5212](#)
- Make copying locks impossible. [References: pull request 5209](#)
- Properly truncate trailing bits of EDNS Client Subnet masks. [References: pull request 5320](#)
- Fix regressions in the AXFR rectification code (Kees Monshouwer, Arthur Gautier). [References: pull request 5161, pull request 5083](#)
- Zero the port when creating a netmask from a `ComboAddress`. [References: pull request 5408](#)
- Drop (broken) support for packet-specific SOA replies from backends (Christian Hofstaedtler). [References: pull request 5512](#)
- Fix validation at the exact RRSIG inception or expiration time [References: pull request 5525](#)

- Lookups one level (or more) below apex did confuse `getAuth()` for qytpe DS (Kees Monshouwer). [🔗 References: pull request 5519](#)
- First and last SOA in an AXFR must be identical (Kees Monshouwer). [🔗 References: pull request 5633](#)
- Make the URL in zone info absolute (Christian Hofstaedtler). [🔗 References: #4524, pull request 4526](#)
- Avoid creating fake `DNSPacket` objects just for calling `getAuth()` from API code (Christian Hofstaedtler). [🔗 References: pull request 5516](#)
- Check if the API is read-only on crypto keys methods. [🔗 References: pull request 5589](#)
- Fix `getSOA()` in `luabackend` (@zilopbg). [🔗 References: pull request 5556](#)
- Avoid undefined behaviour in Clang vs. GCC when printing DS records in `pdnsutil`. [🔗 References: pull request 4740](#)
- In `pdnsutil create-slave-zone`, actually add all slaves. [🔗 References: #5124, pull request 5125](#)
- Fix off-by-one in `dnsreplay -packet-limit` [🔗 References: pull request 5303](#)
- Fix that `pdnsutil edit-zone` complains about `auth=1` problems on all data. [🔗 References: pull request 5610](#)
- Do not corrupt data supplied by other backends in `getAllDomains` (Christian Hofstaedtler). [🔗 References: #4328, pull request 4650](#)
- Reconnect to the server if the My/Pg connection has been closed. [🔗 References: #3824, #5005, pull request 5245](#)
- Make statement actually unique (Christian Hofstaedtler). [🔗 References: #4928, pull request 4929](#)
- Add missing query for last key insert id in the `goracle` backend (Aki Tuomi). [🔗 References: pull request 5506](#)
- Fix `ldap-strict autoptr` feature. [🔗 References: #3165, pull request 4922](#)
- Fix an erroneous `'.'` in `“ip6.arpa”` (@shantikulkarni). [🔗 References: #5091, pull request 5340](#)
- Apply weights consistently during GeoIP lookups (Aki Tuomi). [🔗 References: #4704, pull request 5267](#)
- Fix two problems with `remotebackend` (Aki Tuomi):
 - list method used `domain-id json` parameter, when it was supposed to use `domain_id`
 - NULL `ordname` was not passed as empty string in POST parameters builder, instead it threw an exception[🔗 References: pull request 4997](#)
- Don't copy data around in the Remote Backend when sending and receiving in the Unix Connector. [🔗 References: #5306, pull request 5308](#)

19.2 Changelogs for 4.0.x

19.2.1 PowerDNS Authoritative Server 4.0.5

Released 27th of November 2017

This release fixes PowerDNS Security Advisory [2017-04](#): Missing check on API operations (CVE-2017-15091).

Bug fixes

- [#4650](#): Bindbackend: do not corrupt data supplied by other backends in `getAllDomains` (Christian Hofstaedtler)
- [#4751](#): API: prevent sending nameservers list and zone-level NS in rrsets (Christian Hofstaedtler)

- [#4929](#): gpgsql: make statement names actually unique (Christian Hofstaedtler)
- [#4997](#): Fix remotebackend params (Aki Tuomi)
- [#5051](#): Fix godbc query logging
- [#5125](#): For create-slave-zone, actually add all slaves, and not only first n times
- [#5161](#): Fix a regression in axfr-rectify + test (Arthur Gautier)
- [#5408](#): When making a netmask from a comboaddress, we neglected to zero the port
- [#5599](#): Fix libatomic detection on ppc64
- [#5641](#): Catch DNSName exception in the Zoneparser
- [#5722](#): Publish inactive KSK/CSK as CDNSKEY/CDS
- [#5730](#): Handle AFSDB record separately due to record structure. Fixes [#4703](#) (Johan Jatko)
- [#5678](#): Treat requestor's payload size lower than 512 as equal to 512
- [#5766](#): Correctly purge entries from the caches after a transfer
- [#5777](#): Handle a signing pipe worker dying with work still pending
- [#5815](#): Ignore SOA-EDIT for PRESIGNED zones. Fixes [#5814](#)
- [#5933](#): Check return value for all getTSIGKey calls. Fixes [#5931](#)
- **[#5996](#): Deny cache flush, zone retrieve and notify if the API is RO (Security Advisory [2017-04](#))**

Improvements

- [#4922](#): Fix ldap-strict autoptr feature, including a test
- [#5043](#): mydnsbackend: Add getAllDomains (Aki Tuomi)
- [#5112](#): Stubresolver: Use only `recursor` setting if given
- [#5147](#): LuaWrapper: Allow embedded NULs in strings received from Lua
- [#5277](#): sdig: Clarify that the `ednssubnet` option takes "subnet/mask"
- [#5309](#): Tests: Ensure all required tools are available (Arthur Gautier)
- [#5320](#): PowerDNS sdig does not truncate trailing bits of EDNS Client Subnet mask
- [#5349](#): LuaJIT 2.1: Lua fallback functionality no longer uses Lua namespace
- [#5498](#): Add support for Botan 2.x
- [#5509](#): Ship ldapbackend schema files in tarball (Christian Hofstaedtler)
- [#5518](#): Collection of schema changes (Kees Monshouwer)
- [#5523](#): Fix typo in two log messages (Ruben Kerkhof)
- [#5598](#): Add help text on autodetecting systemd support
- [#5723](#): Use a unique pointer for bind backend's `d_of`
- [#5826](#): Fix some of the issues found by @jpmens

19.2.2 PowerDNS Authoritative Server 4.0.4

Released 23rd of June 2017

This release features a fix for the [ed25519](#) signer. This signer hashed the message before signing, resulting in unverifiable signatures. Also on the Elliptic Curve front, support was added for ED448 (DNSSEC algorithm 16) by using libdecaf.

Bug fixes

- [#5423](#): Do not hash the message in the ed25519 signer (Kees Monshouwer)
- [#5445](#): Make URI integers 16 bits, fixes [#5443](#)
- [#5346](#): configure.ac: Corrects syntax error in test statement on existence of libcrypto_ecdsa (shinsterneck)
- [#5440](#): configure.ac: Fix quoting issue fixes [#5401](#)
- [#4824](#): configure.ac: Check in the detected OpenSSL/libcrypto for ECDSA
- [#5016](#): configure.ac: Check if we can link against libatomic if needed
- [#5341](#): Fix typo in ldapbackend.cc from issue [#5091](#) (shantikulkarni)
- [#5289](#): Sort NSEC record case insensitive (Kees Monshouwer)
- [#5378](#): Make sure NSEC ordernames are always lower case
- [#4781](#): API: correctly take TTL from first record even if we are at the last comment (Christian Hofstaedtler)
- [#4901](#): Fix AtomicCounter unit tests on 32-bit
- [#4911](#): Fix negative port detection for IPv6 addresses on 32-bit
- [#4508](#): Remove support for ‘right’ timezones, as this code turned out to be broken
- [#4961](#): Lowercase the TSIG algorithm name in hash computation
- [#5048](#): Handle exceptions raised by `closesocket()`
- [#5297](#): Don’t leak on signing errors during outgoing AXFR; signpipe stumbles over interrupted rrsets; fix memory leak in gmysql backend
- [#5450](#): TinyCDB backend: Don’t leak a CDB object in case of bogus data

Improvements

- [#5071](#): ODBC backend: Allow query logging
- [#5441](#): Add ED25519 (algo 15) and ED448 (algo 16) support with libdecaf signer (Kees Monshouwer)
- [#5325](#): YaHTTP: Sync with upstream changes
- [#5298](#): Send a notification to all slave servers after every dnsupdate (Kees Monshouwer)
- [#5317](#): Add option to set a global `lua-axfr-script` value (Kees Monshouwer)
- [#5130](#): dnsreplay: Add `--source-ip` and `--source-port` options
- [#5085](#): calidns: Use the correct socket family (IPv4 / IPv6)
- [#5170](#): Add an option to allow AXFR of zones with a different (higher/lower) serial (Kees Monshouwer)
- [#4622](#): API: Make trailing dot handling consistent with pdnsutil (Tuxis Internet Engineering)
- [#4762](#): SuffixMatchNode: Fix insertion issue for an existing node
- [#4861](#): Do not resolve the NS-records for NOTIFY targets if the “only-notify” whitelist is empty, as a target will never match an empty whitelist.
- [#5378](#): Improve the AXFR DNSSEC freshness check; Ignore NSEC3PARAM metadata in an unsigned zone
- [#5297](#): Create additional `reuseport` sockets before dropping privileges; remove transaction in pgpsql backend

19.2.3 PowerDNS Authoritative Server 4.0.3

Released January 17th 2017

This release fixes an issue when using multiple backends, where one of the backends is the BIND backend. This regression was introduced in 4.0.2.

Bug fix

- [#4905](#): Revert “auth: In Bind2Backend::lookup(), use the zoneId when we have it”

19.2.4 PowerDNS Authoritative Server 4.0.2

Released January 13th 2017

This release fixes PowerDNS Security Advisories [2016-02](#), [2016-03](#), [2016-04](#) and [2016-05](#) and includes a fix for a memory leak in the Postgresql backend.

Bug fixes

- [commit f61af48](#): Don't parse spurious RRs in queries when we don't need them (Security Advisory [2016-02](#))
- [commit 592006d](#): Don't exit if the webserver can't accept a connection (Security Advisory [2016-03](#))
- [commit e85acc6](#): Check TSIG signature on IXFR (Security Advisory [2016-04](#))
- [commit 3b1e4a2](#): Correctly check unknown record content size (Security Advisory [2016-05](#))
- [commit 9ecbf02](#): ODBC backend: actually prepare statements
- [commit a4d607b](#): Fix incorrect length check in DNSName when extracting qtype or qclass
- [commit c816fe3](#): Fix a possible memory leak in the webserver
- [#4287](#): Better handling of invalid serial
- [#4306](#): Limit size of mysql cell to 128 kilobytes
- [#4314](#): Overload fix: make overload-queue-length work as intended again, add test for it.
- [#4317](#): Improve root-zone performance
- [#4319](#): pipe: SERVFAIL when needed
- [#4360](#): Make sure mariadb (mysql on centos/rhel) is started before pdns (42wim)
- [#4387](#): ComboAddress: don't allow invalid ports
- [#4459](#): Plug memory leak in postgresql backend (Christian Hofstaedtler)
- [#4544](#): Fix a stack-based off-by-one write in the HTTP remote backend
- [#4755](#): calidns: Don't crash if we don't have enough 'unknown' queries remaining

Additions and Enhancements

- [commit 1238e06](#): disable negative getSOA caching if the negcache_ttl is 0 (Kees Monshouwer)
- [commit 3a0bde](#), [commit 8c879d4](#), [commit 8c03126](#), [commit 5656e12](#) and [commit c1d283d](#): Improve PacketCache cleaning (Kees Monshouwer)
- [#4261](#): Strip trailing dot in PTR content (Kees Monshouwer)
- [#4269](#): contrib: simple bash completion for pdnsutil (j0ju)

- [#4272](#): Bind backend: update status message on reload, keep the existing zone on failure
- [#4274](#): report DHCID type (Kees Monshouwer)
- [#4310](#): Fix build with LibreSSL, for which OPENSSL_VERSION_NUMBER is irrelevant
- [#4323](#): Speedup DNSName creation
- [#4335](#): fix TSIG for single thread distributor (Kees Monshouwer)
- [#4346](#): change default for any-to-tcp to yes (Kees Monshouwer)
- [#4356](#): Don't look up the packet cache for TSIG-enabled queries
- [#4403](#): (auth) Fix build with OpenSSL 1.1.0 final (Christian Hofstaedtler)
- [#4442](#): geoipbackend: Fix minor naming issue (Aki Tuomi)
- [#4454](#): pdnsutil: create-slave-zone accept multiple masters (Hannu Ylitalo)
- [#4541](#): Backport of [#4542](#): API: search should not return ENTs (Christian Hofstaedtler)
- [#4754](#): In `Bind2Backend::lookup()`, use the `zoneId` when we have it

19.2.5 PowerDNS Authoritative Server 4.0.1

Released July 29th 2016

This release fixes two small issues and adds a setting to limit AXFR and IXFR sizes, in response to [CVE-2016-6172](#).

Bug fixes

- [#4126](#) Wait for the connection to the carbon server to be established
- [#4206](#) Don't try to deallocate empty PG statements
- [#4245](#) Send the correct response when queried for an NSEC directly (Kees Monshouwer)
- [#4252](#) Don't include bind files if length ≤ 2 or $> \text{sizeof}(\text{filename})$
- [#4255](#) Catch `runtime_error` when parsing a broken MNAME

Improvements

- [#4044](#) Make `DNSPacket` return a `ComboAddress` for local and remote (Aki Tuomi)
- [#4056](#) OpenSSL 1.1.0 support (Christian Hofstaedtler)
- [#4169](#) Fix typos in a logmessage and exception (Christian Hofstaedtler)
- [#4183](#) pdnsutil: Remove checking of ctime and always diff the changes (Hannu Ylitalo)
- [#4192](#) dnsreplay: Only add Client Subnet stamp when asked
- [#4250](#) Use `toLogString()` for `ringAccount` (Kees Monshouwer)

Additions

- [#4133](#) Add limits to the size of received {A,I}XFR ([CVE-2016-6172](#))
- [#4142](#) Add used filedescriptor statistic (Kees Monshouwer)

19.2.6 PowerDNS Authoritative Server 4.0.0

Released July 11th 2016

PowerDNS Authoritative Server 4.0.0 is part of the great 4.x “Spring Cleaning” of PowerDNS which lasted through the end of 2015.

As part of the general cleanup and improvements, we did the following:

- Moved to C++ 2011, a cleaner more powerful version of C++ that has allowed us to [improve the quality of implementation](#) in many places.
- Implemented dedicated infrastructure for dealing with DNS names that is fully “DNS Native” and needs less escaping and unescaping.
- All backends derived from the Generic SQL backend use [prepared statements](#).
- Both the server and `pdns_control` do the right thing when `chroot`'ed.

In addition to this cleanup, 4.0.0 brings the following new features:

- A revived ODBC backend ([godbc](#)).
- A revived LDAP backend ([ldap](#)).
- Support for [CDS/CDNSKEY](#) and [RFC 7344](#) key-rollovers.
- Support for the [ALIAS](#) record.
- The webserver and API are no longer marked experimental.
 - The API-path has moved to `/api/v1`
- DNSUpdate is no longer experimental.
- Default ECDSA (algorithms 13 and 14) support without external dependencies.
- Experimental support for ed25519 DNSSEC signatures (when compiled with libsodium support).
- IXFR consumption support.
- Many new `pdnsutil` commands
 - `help` command now produces the help
 - Warns if the configuration file cannot be read
 - Does not check disabled records with `check-zone` unless verbose mode is enabled
 - `create-zone` command creates a new zone
 - `add-record` command to add records
 - `delete-rrset` and `replace-rrset` commands to delete and add rrsets
 - `edit-zone` command that spawns `$EDITOR` with the zone contents in zonefile format regardless of the backend used ([blogpost](#))

The following backend have been dropped in 4.0.0:

- LMDB.
- Geo (use the improved [GeoIP](#) instead).

Important changes:

- `pdnssec` has been renamed to `pdnsutil`
- PowerDNS Authoritative Server now listens by default on all IPv6 addresses.
- The default for `pdnsutil secure-zone` has been changed from 1 2048 bit RSA KSK and 1 1024 bit RSA ZSK to a single 256 bit ECDSA (algorithm 13, ECDSAP256SHA256) key.
- Several superfluous queries have been dropped from the SQL backend, if you use a non-standard SQL schema, please review the new defaults

- `insert-ent-query`, `insert-empty-non-terminal-query`, `insert-ent-order-query` have been replaced by one query named `insert-empty-non-terminal-order-query`
- `insert-record-order-query` has been dropped, `insert-record-query` now sets the `ordname` (or `NULL`)
- `insert-slave-query` has been dropped, `insert-zone-query` now sets the type of zone
- Crypto++ and mbedTLS support is dropped, these are replaced by OpenSSL
- The INCEPTION, INCEPTION-WEEK and EPOCH SOA-EDIT metadata values are marked as deprecated and will be removed in 4.1

The final release has the following bug fixes compared to rc2:

- [#4071](#) Abort on backend failures at startup and retry while running (Kees Monshouwer)
- [#4099](#) Don't leak TCP connection descriptor if `pthread_create()` failed
- [#4137](#) `sqlite3`: Check whether foreign keys should be turned on (Aki Tuomi)

And the following improvements:

- [#3051](#) Better error message for unfound new slave domains
- [#4123](#) `check-zone`: warn on mismatch between algo and NSEC mode

19.2.7 PowerDNS Authoritative Server 4.0.0-rc2

Released June 29th 2016

note: rc1 was tagged in git but never officially released. Kees Monshouwer discovered an issue in the `gmysql` backend that would terminate the daemon on a connection error, this fixed in rc2.

This Release Candidate adds IXFR consumption and fixes some issues with prepared statements:

- [#3937](#) GSQL: use lazy prepared statements (Aki Tuomi)
- [#3949](#) Implement IXFR-based slaving for Authoritative, fix duplicate AXFRs
- [#4066](#) Don't die on a mysql timeout (Kees Monshouwer)

Other improvements:

- [#4061](#) Various fixes, a MySQL-query fix that improves performance and one that allows shorter best matches in `getAuth()`
- [#3962](#) Fix OpenBSD support
- [#3972](#) API: change PATCH/PUT on zones to return 204 No Content instead of full zone (Christian Hofstaedtler)
- [#3917](#) Remotebackend: Add `getAllDomains` call (Aki Tuomi)

Bug fixes and changes:

- [#3998](#) remove `mysql::isOurDomain` for now (Kees Monshouwer)
- [#3989](#) Fix usage of `std::distance()` in `DNSName::isPartOf()`
- [#4001](#) re enable `validDNSName()` check (Kees Monshouwer)
- [#3930](#) Have `pdns_control` bind-add-zone check for zonefile
- [#3400](#) Fix building on OpenIndiana
- [#3961](#) Allow building on CentOS 6 i386
- [#3940](#) auth: Don't build `dnsbulktest` and `dnstcpbench` if boost is too old, fixes building on CentOS 6
- [#3931](#) Rename `notify` to `pdns_notify` (Christian Hofstaedtler)

19.2.8 PowerDNS Authoritative Server 4.0.0-beta1

Released May 27th 2016

This release features several small fixes and deprecations.

Improvements and Additions

- [#3851](#) Disable algorithm 13 and 14 if OpenSSL does not support ecdsa or the required curves (Kees Monshouwer)
- [#3857](#) Add simple stubquery tool for testing the stubresolver
- [#3859](#) build scripts: Stop patching config-dir in pdns.conf (Christian Hofstaedtler)
- [#3872](#) Add support for multiple carbon servers
- [#3901](#) Add support for virtual hosting with systemd

Bug fixes

- [#3856](#) Deal with unset name in nproxy replies

19.2.9 PowerDNS Authoritative Server 4.0.0-alpha3

Released May 11th 2016

Notable changes since 4.0.0-alpha2

- [#3415](#) pdnsutil: add clear-zone command
- [#3586](#) Remove send-root-referral option
- [#3578](#) Add disable-syslog option
- [#3733](#) ALIAS improvements: DNSSEC and optional on-AXFR expansion of records
- [#3764](#) Notify support for systemd
- [#3807](#) Add TTL settings for DNSSECKeeper's caches

Bug fixes

- [#3553](#) pdnsutil: properly show key sizes for presigned zones in show-zone
- [#3507](#) webserver: mask out the api-key setting (Christian Hofstaedtler)
- [#3580](#) bindbackend: set domain in list() (Kees Monshouwer)
- [#3595](#) pdnsutil: add NS record without trailing dot with create-zone
- [#3653](#) Allow tabs as whitespace in zonefiles
- [#3666](#) Restore recycle backend behaviour (Kees Monshouwer)
- [#3612](#) Prevent segfault in PostgreSQL backend
- [#3779](#), [#3768](#), [#3766](#), [#3783](#) and [#3789](#) DNSName and other hardening improvements
- [#3784](#) fix SOA caching with multiple backends (Kees Monshouwer)
- [#3827](#) Force NSEC3PARAM algorithm to 1, fixes validation issues when set to not 1

Improvements

- [#3637](#), [#3678](#), [#3740](#) Correct root-zone slaving and serving (Kees Monshouwer and others)
- [#3495](#) API: Add discovery endpoint (Christian Hofstaedtler)
- [#3389](#) pdnsutil: support chroot
- [#3596](#) Remove botan-based ecdsa and rsa signers (Kees Monshouwer)
- [#3478](#), [#3603](#), [#3628](#) Various build system improvements (Ruben Kerkhof)
- [#3621](#) Always lowercase when inserting into the database
- [#3651](#) Rename PUBLISH_* to PUBLISH-* domainmetadata
- [#3656](#) API: clean up cryptokeys resource (Christian Hofstaedtler)
- [#3632](#) pdnsutil: Fix exit statuses to constants and return 0 when success (saltsa)
- [#3655](#) API: Fix set-ptr to honor SOA-EDIT-API (Christian Hofstaedtler)
- [#3720](#) Many fixes for dnswasher (Robert Edmonds)
- [#3707](#), [#3788](#) Make MySQL timeout configurable (Kees Monshouwer and Brynjar Eide)
- [#3806](#) Move key validity check out of `fromISCMAP()`, improves DNSSEC performance
- [#3820](#) pdnsutil load-zone: ignore double SOA

19.2.10 PowerDNS Authoritative Server 4.0.0-alpha2

Released February 25th 2016

Notable changes since 4.0.0-alpha1

- [#3037](#) Remove superfluous `gsqL` queries and stop relying on schema defaults
- [#3176](#), [#3139](#) OpenSSL support (Christian Hofstaedtler and Kees Monshouwer)
- [#3128](#) ECDSA support to DNSSEC infra via OpenSSL (Kees Monshouwer)
- [#3281](#), [#3283](#), [#3363](#) Remove Crypto++ and mbedTLS support
- [#3298](#) Implement pdnsutil `create-zone` `zone nsname`, `add-record`, `delete-rrset`, `replace-rrset`
- [#3407](#) API: Permit wildcard manipulation (Aki Tuomi)
- [#3230](#) API: drop JSONP, add web security headers (Christian Hofstaedtler)
- [#3428](#) API: Fix zone/records design mistake (Christian Hofstaedtler)
 - **Note:** this is a breaking change from alpha1, please review the *API documentation* <..[httpapi](#)>

Bug fixes

- [#3124](#) Fix several bugs with introduced with the change to a single signing key (e.g. the SEP bit is set on these single keys)
- [#3151](#) Catch `DNSName` build errors in `dynhandler` (Christian Hofstaedtler)
- [#3264](#) GeoIP backend: Use correct id numbers for domains (Aki Tuomi)
- [#3271](#) ZoneParser: Throw `PDNSException` on too many SOA data elements
- [#3302](#) Fix `bindbackend`'s `feedRecord` to handle being slave for the root
- [#3399](#) Report OpenSSL RSA keysize in bits (Kees Monshouwer)

Improvements

- [#3119](#) Show DNSSEC keys for slaved zone (Aki Tuomi)
- [#3255](#) Don't log authentication errors before sending HTTP basic auth challenge (Jan Broer)
- [#3338](#) Add weight feature to GeoIP backend (Aki Tuomi)
- [#3364](#) Shrink PacketID by 10% by eliminating padding. (Andrew Nelles)
- [#3443](#) Many speedup and correctness fixes

19.2.11 PowerDNS Authoritative Server 4.0.0-alpha1

Released December 24th 2015

19.3 Changelogs for 3.x and older

These changelogs are included for historical purposes. Broken links may exist.

19.3.1 PowerDNS Authoritative Server 3.4.9

Released 17th of May 2016

This is a minor bugfix and performance release. Two contributions by Kees Monshouwer make 3.4.9 fully compatible with the new single key ECDSA default that is coming in version 4.0.0.

Changes since 3.4.8:

- [commit 4627ea0](#), [commit 8350828](#): use OpenSSL for ECDSA signing where available (Kees Monshouwer)
- [commit 558ff84](#): allow common signing key (Kees Monshouwer)
- [commit 280d665](#): Add a disable-syslog setting
- [commit 58d6ab6](#): fix SOA caching with multiple backends (Kees Monshouwer)
- [commit e9e413f](#), [commit 6af4652](#): whitespace-related zone parsing fixes [ticket #3568](#)
- [commit 7473a5e](#): bindbackend: fix, set domain in list() (Kees Monshouwer)

19.3.2 PowerDNS Authoritative Server 3.4.8

Released 3rd of February 2016

This is a small bugfix release. Additionally, the deb/RPM packages on downloads.powerdns.com (those with -static in the name) for 3.4.8 have been built against Botan 1.10.11 instead of Botan 1.10.3 like previous packages. Please see [the Botan Security page](#) for more information on the fixes in Botan 1.10.11. As a PowerDNS user, these issues only affect you if you ran our -static packages *and* allowed your users to upload private keys to your configuration.

Changes since 3.4.7:

- [commit edfa60a](#): Use AC_SEARCH_LIBS (Ruben Kerkhof)
- [commit 7b7a3af](#): Check for inet_aton in libresolv (Ruben Kerkhof)
- [commit 9322aee](#): Remove hardcoded -lresolv, -lnsl and -lsocket (Ruben Kerkhof)
- [commit 23d26d8](#): pdnssec: don't check disabled records (Pieter Lexis)
- [commit ce92ff1](#): pdnssec: check all records (including disabled ones) only in verbose mode (Kees Monshouwer)

- [commit f745312](#): trailing dot in DNAME content (Kees Monshouwer)
- [commit ed02761](#): Fix luabackend compilation on FreeBSD i386 (RvdE)
- [commit 07ea6ac](#): silence g++ 6.0 warnings and error (Kees Monshouwer)
- [commit c6077b1](#): add gcc 5.3 and 6.0 support to boost.m4 (Kees Monshouwer)

19.3.3 PowerDNS Authoritative Server 3.4.7

Released 3rd of November 2015

This is a security release fixing [Security Advisory 2015-03](#)

Bug fixes:

- [commit b0c04ba](#): Ignore invalid/empty TKEY and TSIG records (Christian Hofstaedtler)
- [commit 8044a5d](#): Don't reply to truncated queries (Christian Hofstaedtler)
- [commit 6a65ae9](#): don't log out-of-zone ents during AXFR in (Kees Monshouwer)
- [commit 416d252](#): Prevent XSS by escaping user input. Thanks to Pierre Jaury and Damien Cauquil at Sysdream for pointing this out.
- [commit df76bda](#): Handle NULL and boolean properly in gPGSql (Aki Tuomi)
- [commits b998fc0](#), [88516fd](#), [ef80925](#), [4549a72](#): Improve negative caching (Kees Monshouwer)
- [commit be27a9c](#): Do not divide timeout twice (Aki Tuomi)
- [commits ca1d29c](#), [df2d20a](#), [2358eea](#): Correctly sort records with a priority.

Improvements:

- [commits 791bc37](#), [e3301ca](#), [9862779](#), [b59a7e3](#), [4ca7a06](#), [7736530](#), [69ea1a6](#): Direct query answers and correct zone-rectification in the GeoIP backend (Aki Tuomi)
- [commits 83e0e53](#), [0ff3037](#), [9910908](#) Use token names to identify PKCS#11 keys (Aki Tuomi)
- [commit a3801b2](#): Fix typo in an error message (Arjen Zonneveld)
- [commit d33ba8e](#): limit NSEC3 iterations in bindbackend (Kees Monshouwer)
- [commit 0acca87](#): Initialize minbody (Aki Tuomi)

New features:

- [commits 4d51e96](#), [6873a07](#), [b972356](#), [46294b5](#), [6277b14](#): OPENPGPKEY record-type (James Cloos and Kees Monshouwer)
- [commit ec0ded7](#): add global soa-edit settings (Kees Monshouwer)

19.3.4 PowerDNS Authoritative Server 3.4.6

Released 28th of August 2015

This is a security release fixing [Security Advisory 2015-02](#)

Bug fixes:

- [commits c849701](#) and [8c91e2c](#): Avoid superfluous backend recycling
- [commits 463fcff](#), [0fc08e8](#), [0fbe69c](#), [1a6af1c](#) and [07f69d3](#): Removal of dnstest from the authoritative server distribution (Kees Monshouwer among others).
- [commits 5cfea4c](#) and [ef011d9](#): Add EDNS unknown version handling and tests EDNS unknown version handling (Aki Tuomi)

Improvements:

- commits [88dd8a7](#) and [dc6c63d](#): Update YaHTTP to v0.1.7 (Aki Tuomi)
- [commit 0a344bc](#): Make trailing/leading spaces stand out in `pdnssec check_zone`
- commits [2e982ad](#) and [09bec1f](#): GCC 5.2 support and sync boost.m4 macro with upstream (Kees Monshouwer among others)
- [commit 1ad4e44](#): Log answer packets only if log-dns-details is enabled (Kees Monshouwer)

19.3.5 PowerDNS Authoritative Server 3.3.3

Released 9th of June 2015

This is a security release fixing [Security Advisory 2015-01](#)

Bug fixes:

- [commit a0a1482](#): Limit the maximum length of a qname

19.3.6 PowerDNS Authoritative Server 3.4.5

Released 9th of June 2015

This is a security release fixing [Security Advisory 2015-01](#)

Bug fixes:

- [commit ffaae2b](#): be careful reading empty lines in our config parser and prevent integer overflow.
- [commit 8e30209](#): prevent crash after `^^list-modules` (Ruben Kerkhof)
- [commit 6cf71cf](#): Limit the maximum length of a qname

Improvements:

- [commit 28ba3fc](#), [commit 61b316f](#): Support `/etc/default` for our debian/ubuntu packages (Aki Tuomi)
- [commit d80e2b6](#): Detect GCC 5.1 for boost (Ruben Kerkhof)
- [commit 68b4834](#), [commit 3b14545](#), [commit 2356d5c](#), [commit 432808b](#): Various PKCS#11 fixes and improvements (Aki Tuomi)
- [commit bf357ff](#), [commit 2433d2e](#), [commit 8fabf4d](#): Fix Coverity issues (Aki Tuomi)
- [commit 5d02d01](#), [commit 7798aa3](#), [commit 9f6e411](#), [commit e25a09c](#): Fix building on OpenBSD (Florian Obser and Ruben Kerkhof)
- [commit 5c8bba2](#): Look for mbedtls before polarssl (Ruben Kerkhof)
- [commit 5abd150](#): Let pkg-config determine botan dependency libs (Ruben Kerkhof)
- [commit ba4d623](#): kill some further mallocs and add note to remind us not to add them back
- [commit 50346d8](#): Move remotebackend-unix test socket to testsdir (Aki Tuomi)
- [commit 32e9512](#): Defer launch of coprocess until first question (Aki Tuomi)
- [commit d9b3ecb](#), [commit 561373e](#): pdnssec: check for glue and delegations in parent zones (Kees Monshouwer)

19.3.7 PowerDNS Authoritative Server 3.3.2

Released 1st of May, 2015

Among other bug fixes and improvements (as listed below), this release incorporates a fix for CVE-2015-1868, as detailed in [PowerDNS Security Advisory 2015-01](#)

If you are running DNSSEC with version 3.3.1 or below, and you cannot currently upgrade to 3.4.4, please consider upgrading to 3.3.2; it has a lot of improvements and bug fixes and tremendously increases compliance.

We want to explicitly thank Kees Monshouwer for digging up all the DNSSEC improvements and porting them back to this release.

When upgrading, please run “`pdnssec rectify-all-zones`” and trigger an AXFR for all DNSSEC zones to make sure you benefit from all the compliance improvements present in this version.

Security fixes:

- [commit 9df4944](#): import CVE-2015-1868 patch (Peter van Dijk)
- [commit dbedfc5](#): kill some further mallocs and add note to remind us not to add them back (bert hubert)

Improvements:

- [commit d0af589](#) , [commit c45b6db](#) , [commit 88c1f21](#) , [commit 2a4c620](#) , [commit 4a4597e](#) , [commit 9fa7373](#) , [commit 8115a83](#): implement security polling for auth
- [commit 5bbd868](#): import suck() from master (Kees Monshouwer)
- [commit 194f4d2](#): respond REFUSED instead of NOERROR for “unknown zone” situations (Peter van Dijk)
- [commit 55b0653](#): set AA on CNAME into referral, fixes [ticket #589](#) (Peter van Dijk)
- [commit 71232aa](#): update l.root ip (Kees Monshouwer)

Bug fixes:

- [commit 88c52fe](#): make makeRelative() case insensitive (Kees Monshouwer)

DNSSEC improvements:

- [commit b3dec9c](#): change default for add-superfluous-nsec3-for-old-bind config option (Kees Monshouwer)
- [commit 017a78b](#): limit the number of NSEC3 iterations RFC5155 10.3 (Kees Monshouwer)
- [commit d768d7f](#): NSEC3 and related RRSIGS are not part of the dnstree (Kees Monshouwer)
- [commit 3a36a1c](#): import bindbackend rectify code from master (Kees Monshouwer)
- [commit 1ee7e22](#): limit mode 0 closest provable encloser to optout (Kees Monshouwer)
- [commit bbc0bc5](#): fix for errata 3441 of RFC5155 (Kees Monshouwer)
- [commit e8bfa7b](#): allow covering NSEC3 record in NODATA response (Kees Monshouwer)
- [commit f0b3b24](#): return NOTIMP for direct RRSIG request (Kees Monshouwer)
- [commit c79addc](#): import pdnssec checkZone() from master (Kees Monshouwer)
- [commit 2f1fec7](#): import pdnssec rectifyZone() from master (Kees Monshouwer)

19.3.8 PowerDNS Authoritative Server 3.4.4

Released 23rd of April, 2015

Warning: Version 3.4.4 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Additionally, if you are coming from any 3.x version (including 3.3.1), there is a mandatory SQL schema upgrade. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Among other bug fixes and improvements (as listed below), this release incorporates a fix for CVE-2015-1868, as detailed in [PowerDNS Security Advisory 2015-01](#)

Bug fixes:

- [commit ac3ae09](#): fix rectify-(all)-zones for mixed case domain names
- [commit 2dea55e](#), [commit 032d565](#), [commit 55f2dbf](#): fix CVE-2015-1868

- [commit 21cdbe5](#): Blocking IO in busy-wait for remote backend (Wieger Opmeer)
- [commit cc7b2ac](#): fix double dot for root MX/SRV in bind slave zone files (Kees Monshouwer)
- [commit c40307b](#): Properly lock lmdb database, fixes [ticket #1954](#) (Aki Tuomi)
- [commit 662e76d](#): Fix segfault in zone2lmdb (Ruben Kerkhof)

New Features:

- [commit 5ae212e](#): pdnssec: warn for insecure wildcards in opt-out zones
- [commits cd3f21c](#), [8b582f6](#), [0b7e766](#), [f743af9](#), [dcde3c8](#) and [f12fcf7](#): TKEY record type (Aki Tuomi)
- [commits 0fda1d9](#), [3dd139d](#), [ba146ce](#), [25109e2](#), [c011a01](#), [0600350](#), [fc96b5e](#), [4414468](#), [c163d41](#), [f52c7f6](#), [8d56a31](#), [7821417](#), [ea62bd9](#), [c5ababd](#), [91c8351](#) and [073ac49](#): Many PKCS#11 improvements (Aki Tuomi)
- [commits 6f0d4f1](#) and [5eb33cb](#): Introduce xfrBlobNoSpaces and use them for TSIG (Aki Tuomi)

Improvements:

- [commit e4f48ab](#): allow “pdnssec set-nsec3 ZONE” for insecure zones; this saves on one rectify when securing a NSEC3 zone
- [commits cce95b9](#), [e2e9243](#) and [e82da97](#): Improvements to the config-file parsing (Aki Tuomi)
- [commit 2180e21](#): postgresql check should not touch LDFLAGS (Ruben Kerkhof)
- [commit 0481021](#): Log error when remote cannot do AXFR (Aki Tuomi)
- [commit 1ecc3a5](#): Speed improvements when AXFR is disabled (Christian Hofstaedtler)
- [commits 1f7334e](#) and [b17799a](#): NSEC3 and related RRSIGS are not part of the dnstree (Kees Monshouwer)
- [commits dd943dd](#) and [58c4834](#): Change ifdef to check for `__GLIBC__` instead of `__linux__` to prevent errors with other libc’s (James Taylor)
- [commit c929d50](#): Try to raise open files before dropping privileges (Aki Tuomi)
- [commit 69fd3dc](#): Add newline to carbon error message on auth (Aki Tuomi)
- [commit 3064f80](#): Make sure we send servfail on error (Aki Tuomi)
- [commit b004529](#): Ship lmdb-example.pl in tarball (Ruben Kerkhof)
- [commit 9e6b24f](#): Allocate TCP buffer dynamically, decreasing stack usage
- [commit 267fdde](#): throw if getSOA gets non-SOA record

19.3.9 PowerDNS Authoritative Server 3.4.3

Warning: Version 3.4.3 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Additionally, if you are coming from any 3.x version (including 3.3.1), there is a mandatory SQL schema upgrade. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Released March 2nd, 2015

Find the downloads on our [download page](#).

Bug fixes:

- [commit ceb49ce](#): pdns_control: exit 1 on unknown command (Ruben Kerkhof)
- [commit 1406891](#): evaluate KSK ZSK pairs per algorithm (Kees Monshouwer)
- [commit 3ca050f](#): always set di.notified_serial in getAllDomains (Kees Monshouwer)
- [commit d9d09e1](#): pdns_control: don’t open socket in /tmp (Ruben Kerkhof)

New features:

- [commit 2f67952](#): Limit who can send us AXFR notify queries (Ruben Kerkhof)

Improvements:

- [commit d7bec64](#): respond REFUSED instead of NOERROR for “unknown zone” situations
- [commit ebeb9d7](#): Check for Lua 5.3 (Ruben Kerkhof)
- [commit d09931d](#): Check compiler for relro support instead of linker (Ruben Kerkhof)
- [commit c4b0d0c](#): Replace PacketHandler with UeberBackend where possible (Christian Hofstaedtler)
- [commit 5a85152](#): PacketHandler: Share UeberBackend with DNSSECKeeper (Christian Hofstaedtler)
- [commit 97bd444](#): fix building with GCC 5

Experimental API changes (Christian Hofstaedtler):

- [commit ca44706](#): API: move shared DomainInfo reader into it’s own function
- [commit 102602f](#): API: allow writing to domains.account field
- [commit d82f632](#): API: read and expose domain account field
- [commit 2b06977](#): API: be more strict when parsing record contents
- [commit 2f72b7c](#): API: Reject unknown types (TYPE0)
- [commit d82f632](#): API: read and expose domain account field

19.3.10 PowerDNS Authoritative Server 3.4.2

Warning: Version 3.4.2 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Additionally, if you are coming from any 3.x version (including 3.3.1), there is a mandatory SQL schema upgrade. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Released February 3rd, 2015

Find the downloads [on our download page](#).

This is a performance and bugfix update to 3.4.1 and any earlier version. For high traffic setups, including those using DNSSEC, upgrading to 3.4.2 may show tremendous performance increases.

A list of changes since 3.4.1 follows.

Improvements:

- [commit 73004f1](#): implement CORS for the HTTP API
- [commit 4d9c289](#): qtype is now case insensitive in API and database
- [commit 13af5d8](#), [commit 223373a](#), [commit 1d5a68d](#), [commit 705a73f](#), [commit b418d52](#): Allow (optional) PIE hardening
- [commit 2f86f20](#): json-api: remove priority from json
- [commit cefcf9f](#): backport remotebackend fixes
- [commit 920f987](#), [commit dd8853c](#): Support Lua 5.3
- [commit 003aae5](#): support single-type ZSK signing
- [commit 1c57e1d](#): Potential fix for [ticket #1907](#), we now try to trigger libgcc_s.so.1 to load before we chroot. I can’t reproduce the bug on my local system, but this “should” help. Seriously.
- [commit 031ab21](#): update polarssl to 1.3.9

Bug fixes:

- [commit 60b2b7c](#), [commit d962fbc](#): refuse overly long labels in names
- [commit a64fd6a](#): auth: limit long version strings to 63 characters and catch exceptions in secpoll
- [commit fa52e02](#): pdnssec: fix ttl check for RRSIG records

- [commit 0678b25](#): fix up latency reporting for sub-millisecond latencies (would clip to 0)
- [commit d45c1f1](#): make sure we don't throw an exception on "pdns_control show" of an unknown variable
- [commit 63c8088](#): fix startup race condition with carbon thread already trying to broadcast uninitialized data
- [commit 796321c](#): make qsize-q more robust
- [commit 407867c](#): mind04 discovered we count corrupt packets and EAGAIN situations as validly received packets, skewing the udp questions/answers graphs on auth.
- [commit f06d069](#): make latency & qsize reporting 'live'. Plus fix that we only reported the qsize of the first distributor.
- [commit 2f3498e](#): fix up statbag for carbon protocol and function pointers
- [commit 0f2f999](#): get priority from table in Lua axfrfilter; fixes [ticket #1857](#)
- [commit 96963e2](#), [commit bbcbbbe](#), [commit d5c9c07](#): various backends: fix records pointing at root
- [commit e94c2c4](#): remove additional layer of trailing . stripping, which broke MX records to the root in the BIND backend. Should close [ticket #1243](#).
- [commit 8f35ba2](#): api: use uncached results for getKeys()
- [commit c574336](#): read ALLOW-AXFR-FROM from the backend with the metadata

Minor changes:

- [commit 1e39b4c](#): move manpages to section 1
- [commit b3992d9](#): secpoll: Replace ~ with _
- [commit 9799ef5](#): only zones with an active ksk are secure
- [commit d02744f](#): api: show keys for zones without active ksk

New features:

- [commit 1b97ba0](#): add signatures metric to auth, so we can plot signatures/second
- [commit 92cef2d](#): pdns_control: make it possible to notify all zones at once
- [commit f648752](#): JSON API: provide flush-cache, notify, axfr-retrieve
- [commit 02653a7](#): add 'bench-db' to do very simple database backend performance benchmark
- [commit a83257a](#): enable callback based metrics to statbas, and add 5 such metrics: uptime, sys-msec, user-msec, key-cache-size, meta-cache-size, signature-cache-size

Performance improvements:

- [commit a37fe8c](#): better key for packetcache
- [commit e5217bb](#): don't do time(0) under signature cache lock
- [commit d061045](#), [commit 135db51](#), [commit 7d0f392](#): shard the packet cache, closing [ticket #1910](#).
- [commit d71a712](#): with thanks to Jack Lloyd, this works around the default Botan allocator slowing down for us during production use.

19.3.11 PowerDNS Authoritative Server 3.4.1

Warning: Version 3.4.1 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Additionally, if you are coming from any 3.x version (including 3.3.1), there is a mandatory SQL schema upgrade. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Released October 30th, 2014

Find the downloads [on our download page](#).

This is a bugfix update to 3.4.0 and any earlier version.

A list of changes since 3.4.0 follows.

- [commit dcd6524](#), [commit a8750a5](#), [commit 7dc86bf](#), [commit 2fda71f](#): PowerDNS now polls the security status of a release at startup and periodically. More detail on this feature, and how to turn it off, can be found in [Security polling](#).
- [commit 5fe6dc0](#): API: Replace HTTP Basic auth with static key in custom header (X-API-Key)
- [commit 4a95ab4](#): Use transaction for pdnssec increase-serial
- [commit 6e82a23](#): Don't empty ordername during pdnssec increase-serial
- [commit 535f4e3](#): honor SOA-EDIT while considering "empty IXFR" fallback, fixes [ticket 1835](#). This fixes slaving of signed zones to IXFR-aware slaves like NSD or BIND.

19.3.12 PowerDNS Authoritative Server 3.4.0

Released September 30th, 2014

This is a performance, feature, bugfix and conformity update to 3.3.1 and any earlier version. It contains a huge amount of work by various contributors, to whom we are very grateful.

Warning: Version 3.4.0 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Additionally, if you are coming from any 3.x version (including 3.3.1), there is a mandatory SQL schema upgrade. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Downloads

Find the downloads [on our download page](#).

A list of changes since 3.3.1 follows.

Changes between RC2 and 3.4.0:

- [commit ad189c9](#), [commit 445d93c](#): also distribute the dnsdist manual page
- [commit b5a276d](#), [commit 0b346e9](#), [commit 74caf87](#), [commit 642fd2e](#): Make sure all backends actually work as dynamic modules
- [commit 14b11c4](#): raise log level on dlerror(), fixes [ticket 1734](#), thanks @James-TR
- [commit 016d810](#): improve postgresql detection during ./configure
- [commit dce1e90](#): DNAME: don't sign the synthesised CNAME
- [commit 25e7af3](#): send empty SERVFAIL after a backend throws a DBException, instead of including useless content

Changes between RC1 and RC2:

- [commit bb6e54f](#): document udp6-queries, udp4-queries, add rd-queries, recursion-unanswered metrics & document. Closes [ticket 1400](#).
- [commit 4a23af7](#): init script: support DAEMON_ARGS; [commit 7e5b3a0](#): init script: ensure socket dir exists
- [commit dd930ed](#): don't import supermaster ips from other accounts
- [commit ed3afdf](#): fall back to central bind if reuseport bind fails; improves [ticket 1715](#)
- [commit 709ca59](#): GeoIP backend implementation. This is a new backend, still experimental!
- [commit bf5a484](#): support EVERY future version of OS X, fixes [ticket 1702](#)

- [commit 4dbaec6](#): Check for `__FreeBSD_kernel__` as per <https://lists.debian.org/debian-bsd/2006/03/msg00127.html>, fixes [ticket 1684](#); [commit 74f389d](#): `__FreeBSD_kernel__` is defined but empty on systems with FreeBSD kernels, breaking compile. Thanks pawal
- [commit 2e6bbd8](#): Catch `PDNSException` in `Signingpiper::helperWorker` to avoid abort
- [commit 0ffd51d](#): improve error reporting on malformed labels
- [commit c48dec7](#): Fix forwarded TSIG message issue
- [commit dad70f2](#): skip `TCP_DEFER_ACCEPT` on platforms that do not have it (like FreeBSD); fixes [ticket 1658](#)
- [commit c7287b6](#): should fix [ticket 1662](#), reloading while checking for domains that need to be notified in BIND, causing lock
- [commit 3e67ea8](#): allow OPT pseudo record type in IXFR query
- [commit a1caa8b](#): webserver: htmlescape VERSION and config name
- [commit df9d980](#): Remove “log-failed-updates” leftover
- [commit a1fe72a](#): Remove unused “soa-serial-offset” option

Changes between 3.3.1 and 3.4.0-RC1 follow.

DNSSEC changes

- [commit bba8413](#): add option (`max-signature-cache-entries`) to limit the maximum number of cached signatures.
- [commit 28b66a9](#): limit the number of NSEC3 iterations (see RFC5155 10.3), with the `max-nsec3-iterations` option.
- [commit b50efd6](#): drop the ‘superfluous NSEC3’ option that old BIND validators need.
- The bindbackend ‘hybrid’ mode was reintroduced by Kees Monshouwer. Enable it with `bind-hybrid`.
- Aki Tuomi contributed experimental PKCS#11 support for DNSSEC key management with a (Soft)HSM.
- Direct RRSIG queries now return NOTIMP.
- [commit fa37777](#): add `secure-all-zones` command to `pdnssec`
- Unrectified zones can now get rectified ‘on the fly’ during outgoing AXFR. This makes it possible to run a hidden signing master without rectification.
- [commit 82fb538](#): AXFR in: don’t accept zones with a mixture of Opt-Out NSEC3 RRs and non-Opt-Out NSEC3 RRs
- Various minor bugfixes, mostly from the unstoppable Kees Monshouwer.
- [commit 0c4c552](#): set non-zero exit status in `pdnssec` if an exception was thrown, for easier automatic usage.
- [commit b8bd119](#): `pdnssec -v show-zone`: Print all keys instead of just entry point keys.
- [commit 52e0d78](#): answer direct NSEC queries without DO bit
- [commit ca2eb01](#): output ZSK DNSKEY records if experimental-direct-dnskey support is enabled
- [commit 83609e2](#): SOA-EDIT: fix INCEPTION-INCREMENT handling
- [commit ac4a2f1](#): AXFR-out can handle secure and insecure NSEC3 optout delegations
- [commit ff47302](#): AXFR-in can handle secure and insecure NSEC3 optout delegations

New features

- DNAME support. Enable with experimental-dname-processing.
- PowerDNS can now send stats directly to Carbon servers. Enable with carbon-server, tweak with carbon-ourname and carbon-interval.
- [commit 767da1a](#): Add list-zone capability to pdns_control
- [commit 51f6bca](#): Add delete-zone to pdnssec.
- The gsql backends now support record comments, and disabling records.
- The new reuseport config option allows setting SO_REUSEPORT, which allows for some performance improvements.
- local-address-nonexist-fail and local-ipv6-nonexist-fail allow pdns to start up even if some addresses fail to bind.
- 'AXFR-SOURCE' in domainmetadata sets the source address for an AXFR retrieval.
- [commit 451ba51](#): Implement pdnssec get-meta/set-meta
- Experimental RFC2136/DNS UPDATE support from Ruben d'Arco, with extensive testing by Kees Monshouwer.
- pdns_control bind-add-zone
- New option bind-ignore-broken-records ignores out-of-zone records while loading zone files.
- pdnssec now has commands for TSIG key management.
- We now support other algorithms than MD5 for TSIG.
- [commit ba7244a](#): implement pdns_control qtypes
- Support for += syntax for options

Bugfixes

- We verify the algorithm used for TSIG queries, and use the right algorithm in signing if there is possible confusion. Plus a few minor TSIG-related fixes.
- [commit ff99a74](#): making *-threads settings empty now yields a default of one instead of zero.
- [commit 9215e60](#): we had a deadly embrace in getUpdatedMasters in bindbackend reimplementation, thanks to Winfried for detailed debugging!
- [commit 9245fd9](#): don't addSuckRequest after supermaster zone creation to avoid one cause of simultaneous AXFR for the same zone
- [commit 719f902](#): fix dual-stack superslave when multiple namservers share a ip
- [commit 33966bf](#): avoid address truncation in doNotifications
- [commit eac85b1](#): prevent duplicate slave notifications caused by different ipv6 address formatting
- [commit 3c8a711](#): make notification queue ipv6 compatible
- [commit 0c13e45](#): make isMaster ip check more tolerant for different ipv6 notations
- Various fixes for possible issues reported by Coverity Scan ([commit f17c93b](#),)
- [commit 9083987](#): don't rely on included polarssl header files when using system polarssl. Spotted by Oden Eriksson of Mandriva, thanks!
- Various users reported pdns_control hangs, especially when using the guardian. We are confident that all causes of these hangs are now gone.
- Decreasing the webserver ringbuffer size could cause crashes.

- [commit 4c89cce](#): nproxy: Add missing `chdir("/")` after `chroot()`
- [commit 016a0ab](#): actually notice timeout during AXFR retrieve, thanks hkraal

REST API changes

- The REST API was much improved and is nearing stability, thanks to Christian Hofstaedtler and others.
- Mark Schouten at Tuxis contributed a zone importer.

Other changes

- Our tarballs and packages now include *.sql schema files for the SQL backends.
- The webserver (including API) now has an ACL (webserver-allow-from).
- Webserver (including API) is now powered by YaHTTP.
- Various autotools usage improvements from Ruben Kerkhof.
- The dist tarball is now bzip2-compressed instead of gzip.
- Various remotebackend updates, including replacing curl with (included) yahttp.
- Dynamic module loading is now allowed on Mac OS X.
- The AXFR ACL (allow-axfr-ips) now defaults to 127.0.0.0/8,::1 instead of the whole world.
- [commit ba91c2f](#): remove unused gpgsql-socket option and document postgres socket usage
- Improved support for Lua 5.2.
- The edns-subnet option code is now fixed at 8, and the edns-subnet-option-numbers option has been removed.
- geobackend now has very limited edns-subnet support - it will use the 'real' remote if available.
- pipebackend ABI v4 adds the zone name to the AXFR command.
- We now [avoid getaddrinfo\(\)](#) as much as possible.
- The packet cache now handles (forwarded) recursive answers better, including TTL aging and respecting allow-recursion.
- [commit ff5ba4f](#): `pdns_server ^help` no longer exits with 1.
- Mark Zealey contributed an experimental LMDB backend. Kees Monshouwer added experimental DNSSEC support to it. Thanks, both!
- [commit 81859ba](#): No longer attempt to answer questions coming in from port 0, reply would not reach them anyhow. Thanks to Niels Bakker and sid3windr for insight & debugging. Closes [ticket 844](#).
- RCodes are now reported in text in various places, thanks Aki.
- Kees Monshouwer set up automatic testing for the oracle and goracle backends, and fixed various issues in them.
- Leftovers of previous support for Windows have been removed, thanks to Kees Monshouwer, Aki Tuomi.
- Bundled PolarSSL has been upgraded to 1.3.2
- PolarSSL replaced previously bundled implementations of AES ([commit e22d9b4](#)) and SHA ([commit 9101035](#))
- bindbackend is now a module
- [commit 14a2e52](#): Use the inet data type for supermasters.ip on postgresql.
- We now send an empty SERVFAIL when a CNAME chain is too long, instead of including the partial chain.
- [commit 3613a51](#): Show built-in features in `^version` output

- [commit 4bd7d35](#): make domainmetadata queries case insensitive
- [commit 088c334](#): output warning message when no to be notified NS's are found
- [commit 5631b44](#): gsqlbackend: use empty defaults for dbname and user; libpq will use the current user name for both by default
- [commit d87ded3](#): implement udp-truncation-threshold to override the previous 1680 byte maximum response datagram size - no matter what EDNS0 said. Plus document it.
- Implement udp-truncation-threshold to override the previous 1680 byte maximum response datagram size - no matter what EDNS0 said.
- Removed settings related to fancy records, as we haven't supported those since version 3.0
- Based on earlier work by Mark Zealey, Kees Monshouwer increased our packet cache performance between 200% and 500% depending on the situation, by simplifying some code in [commit 801812e](#) and [commit 8403ade](#).

19.3.13 PowerDNS Authoritative Server version 3.3.1

Released December 17th, 2013

This is a bugfix update to 3.3.

Downloads

- [Official download page](#)
- [native RHEL5/6 packages from Kees Monshouwer](#)

Changes since 3.3

- direct-dnskey is no longer experimental, thanks Kees Monshouwer & co for extensive testing ([commit e4b36a4](#)).
- Handle signals during poll ([commit 5dde2c6](#)).
- [commit 7538e56](#): Fix zone2{sql,json} exit codes
- [commit 7593c40](#): geobackend: fix possible nullptr deref
- [commit 3506cc6](#): gsqlbackend: don't append empty dbname=/user= values to connect string
- gpgsql queries were simplified through the use of casting ([commit 9a6e39c](#)).
- [commit a7aa9be](#): Replace hardcoded make with variable
- [commit e4fe901](#): make sure to run PKG_PROG_PKG_CONFIG before the first PKG_* usage
- [commit 29bf169](#): fix hmac-md5 TSIG key lookup
- [commit c4e348b](#): fix 64+ character TSIG keys
- [commit 00a7b25](#): Fix comparison between signed and unsigned by using uint32_t for inception on INCEPTION-EPOCH
- [commit d3f6432](#): fix building on os x 10.9, thanks Martijn Bakker.
- We now allow building against Lua 5.2 ([commit bef3000](#), [commit 2bdd03b](#), [commit 88d9e99](#)).
- [commit fa1f845](#): autodetect MySQL 5.5+ connection charset
- When misconfigured using 'right' timezones, a bug in (g)libc gmtime breaks our signatures. Fixed in [commit e4faf74](#) by Kees Monshouwer by implementing our own gmtime_r.
- When sending SERVFAIL due to a CNAME loop, don't uselessly include the CNAMEs ([commit dfd1b82](#)).

- Build fixes for platforms with ‘weird’ types (like s390/s390x): [commit c669f7c](#) (details), [commit 07b904e](#) and [commit 2400764](#).
- Support for += syntax for options, [commit 98dd325](#) and others.
- [commit f8f29f4](#): nproxy: Add missing chdir("/") after chroot()
- [commit 2e6e9ad](#): fix for “missing” libmysqlclient on RHEL/CentOS based systems
- pdnssec check-zone improvements in [commit 5205892](#), [commit edb255f](#), [commit 0dde9d0](#), [commit 07ee700](#), [commit 79a3091](#), [commit 08f3452](#), [commit bcf9daf](#), [commit c9a3dd7](#), [commit 6ebfd08](#), [commit fd53bd0](#), [commit 7eaa83a](#), [commit e319467](#), ,
- NSEC/NSEC3 fixes in [commit 3191709](#), [commit f75293f](#), [commit cd30e94](#), [commit 74baf86](#), [commit 1fa8b2b](#)
- The webserver could crash when the ring buffers were resized, fixed in [commit 3dfb45f](#).
- [commit 213ec4a](#): add constraints for name to pg schema
- [commit f104427](#): make domainmetadata queries case insensitive
- [commit 78fc378](#): no label compression for name in TSIG records
- [commit 15d6ffb](#): pdnssec now outputs ZSK DNSKEY records if experimental-direct-dnskey support is enabled (renamed to direct-dnskey before release!)
- [commit ad67d0e](#): drop cryptopp from static build as libcryptopp.a is broken on Debian 7, which is what we build on
- [commit 7632dd8](#): support polarssl 1.3 externally.
- Remotebackend was fully updated in various commits.
- [commit 82def39](#): SOA-EDIT: fix INCEPTION-INCREMENT handling
- [commit a3a546c](#): add innodb-read-committed option to gmysql settings.
- [commit 9c56e16](#): actually notice timeout during AXFR retrieve, thanks hkraal

19.3.14 PowerDNS Authoritative Server version 3.3

Released on July 5th 2013

This a stability, bugfix and conformity update to 3.2. It improves interoperability with various validators, either through bugfixes or by catering to their needs beyond the specifications.

Warning: Version 3.3 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, 3.1 or 3.2. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Downloads

- [Official download page](#)
- [native RHEL5/6 packages from Kees Monshouwer](#)

Changes between RC2 and final

- pdnssec rectify-zone now refuses to operate on presigned zones, as rectification already happens during incoming transfer. Patch by Kees Monshouwer in [commit 9bd211e](#).
- We now handle zones with a mix of NSEC3 opt-out and non-opt-out ranges correctly during inbound and outbound AXFR. Many thanks to Kees Monshouwer. Code in [commit 5aa7003](#) and [commit d3e7b17](#).

- More remotebackend fixes ([commit 32d4f44](#), [commit 44c2ee8](#), [commit 1fcc7b7](#), [commit 0b1a3b2](#), [commit 9a319b1](#)), thanks Aki Tuomi.
- Some compiler warnings were squashed ([commit ed554db](#)), thanks Morten Stevens.
- Fix broken memory access in LOC parser ([commit 4eec51b](#), [commit bea513c](#)), thanks Aki Tuomi.
- DNSSEC: DS queries at the apex of a zone for which we are not hosting the parent, would wrongly get an ‘unauth NOERROR’. Fixed by Kees Monshouwer in [commit 34479a6](#).

Changes between RC1 and RC2

- Added dnstcpbench tool, by popular demand.
- We always shipped a static tools RPM; we now have a similar Debian package. All packages have been cleaned up a bit, and the binary collections are now consistent between RPM and Deb. New: pass `^enable-tools` to configure to have the tools included in ‘make all’ and ‘make install’.
- [commit 4d2e3f5](#): add selinux policy files
- We would sometimes send a single NULL byte, or nothing at all, instead of an OPT record. Fixed in [commit bf7f822](#), [commit 063076b](#), [commit 90d361d](#).
- [commit 2ee9ba2](#): expand any-to-tcp to direct RRSIG queries
- [commit 5fff084](#), [commit e38ef51](#): drop no-op flag strict-rfc-axfrs, thanks Jelte Jansen.
- [commit f3d8902](#), [commit 7c0b859](#), [commit 5eea730](#): Implement MINFO qtype for better interaction when slaving zones from NSD (that contain MINFO). Thanks to Jelte Jansen.
- [commit 8655a42](#), [commit bf79c6a](#), [commit 38c941b](#): SRV record can have a ‘.’ as final field, from which we would dutifully strip the trailing ‘.’, leaving void, confusing everything. We now remove the trailing ‘.’ in the right place, and not if we are trying to server ‘.’. Again thanks to Jelte & SIDN for catching this.
- [commit 70d5a66](#): improve error message in ill formed unknown record type, thanks Jelte Jansen for reporting.
- [commit 3640473](#): Built in webserver can now listen on IPv6, fixes [ticket 843](#). Also silences some useless messages about timeouts.
- [commit 7db735c](#), [commit d72166c](#): CHANGES BEHAVIOUR: before we launch, check if we can connect to the controlsocket we are about to obliterate. If it works, abort. Fixes [ticket 841](#) and changes standing behaviour. There might be circumstances where PowerDNS now refuses to start, where it previously would. However, starting and making our previous instance mute wasn’t good.
- [commit 9130f9e](#): correctly refuse out-of-zone data in bindbackend, closes [ticket 845](#)
- [commit 3363ef7](#): initialise server-id after all parsing is done, instead of half way through. Fixes situations where server-id was emptied explicitly. Reported by Wouter de Jong
- [commit cd4f253](#): bump boost requirement, thanks Wouter de Jong
- [commit 58cad74](#): Update pdns auth init script so it works on wheezy
- [commit 8714c9c](#): clang fixes by Aki Tuomi, thanks!
- [commit 146601d](#): stretch supermasters.ip for IPv6, thanks Dennis Krul
- [commit 1a5c5f9](#): various remotebackend improvements by Aki Tuomi
- [commit 6ab1a11](#): make sure systemd starts PowerDNS after relevant databases have been started, thanks Morten Stevens.
- [commit 606018f](#), [commit ee5e175](#), [commit c76f6f4](#): check scopeMask of answer packet, not of query packet!
- [commit 2b18bcf](#): Added warning if trailing dot is used, thanks Aki Tuomi.
- [commit 16cf913](#): make superfluous ‘bind’ NSEC3 record optional

New features and important changes since 3.2 (these changes are in RC1 and up)

- [commit 04576ee](#), [commit b0e15c8](#): Implement pdnssec increase-serial, thanks Ruben d'Arco.
- [commit cee857b](#): PowerDNS now sets additional groups while dropping privileges.
- [commit 7796a3b](#): Merge support for include-dir directive, thanks Aki Tuomi!
- [commit d725755](#): make pdns-static Conflict with pdns-server, closes [ticket 640](#)
- [commit c0d5504](#): pdnssec now emits 'INSERT INTO domain ..' queries when running without named.conf, thanks Ruben d'Arco.
- [commit a1d6b0c](#): Older versions of the BIND 9 validating recursor need a superfluous NSEC3 record on positive wildcard responses. We now send this extra NSEC3. Closes [ticket 814](#).
- [commit 07bf35d](#): catch a lot more errors in pdnssec and report them. Fixes [ticket 588](#).
- [commit 032e390](#): make pdnssec exit with 1 on some error conditions, closes [ticket 677](#)
- [commit 4af49b8](#), [commit 4cec6ac](#): add ability to create an 'active' or inactive key using add-zone-key and import-zone-key, plus silenced some debugging. Fixes [ticket 707](#).
- [commit fae4167](#): Compiling against Lua 5.2 (^with-lua=lua5.2) now disables some code used for regression testing, instead of breaking during compile. This means that Lua 5.2 can be used in production.
- [commit abc8f3f](#), [357f6a7](#): Implement the new any-to-tcp option that, when set, always replies with a truncated response (TC=1) to ANY queries, forcing them to use TCP.
- [commit 496073b](#): Since 3.0, pdnssec secure-zone has always generated 3 keys: one KSK and two ZSK, with one ZSK active. For most, if not almost all, users, this inactive ZSK is never used. We now no longer generate this useless ZSK. The resulting smaller DNSKEY RRset improves interoperability with certain validators. Closes [ticket 824](#).
- [commit df55450](#): Non-DNSSEC ANY queries no longer get sent DNSSEC records. This improves interoperability with some old resolvers. Patch by Kees Monshouwer.
- [commit 04b4bf6](#): Merge support for not using opt-out with NSEC3. Many thanks to Kees Monshouwer.
- [commit 8db49a6](#): We now try not to NOTIFY ourselves. In convoluted cases involving REUSE_PORT and binding to 0.0.0.0 and ::, it might be possible that we guess wrong, in which case you can set prevent-self-notification to off.

Important bug fixes

- [commit 63e365d](#): don't mess up encoding when copying qname from question to answer in packetcache. Based on reports&debugging by Jimmy Bergman (sigint), Daniel Norman (Loopia) and the fine people at ISC. This avoids most issues related to BIND 9 erroneously blacklisting PowerDNS for lack of EDNS support.
- [commit 3526186](#): fix backslash handling in TXT parser, includes test. Thanks Jan-Piet Mens.
- [commit 830281f](#), [aef7330](#): Accept chars >127 ('high ASCII') in TXT records, closing [ticket 541](#) and [723](#).
- [commit feef1ec](#): fix missing NSEC3 for secure delegation, thanks Kees Monshouwer, closes [ticket 682](#)
- [commit b61e407](#): around Thursday midnight, during signature rollovers, we would update the SOA serial too early. Fixed by reverting [commit d90efbf](#), adding 7 days margin to inception. Fix by Kees Monshouwer.
- [commit ff64750](#): make sure mixed-case queries get a correct apex NSEC3 type bitmap
- [commit 4b153d8](#): always lowercase next name in NSEC to avoid interop troubles with validators, thanks Marco Davids&Matthijs Mekking.

Other changes

- [commit 49977c6](#): fix bug in boost.m4 where it insists on setting -L, causing useless RPATH in our binaries. Closes [ticket 728](#)
- [commit 62ac758](#): use PolarSSL for MD5 hashing instead of shipping our own copy of md5 hashing code, thanks Aki Tuomi.
- [commit 775acd9](#): give a better error on trying to add nsec3 parameters to a weird zone like “1 0 1 ab” (which indicates that you forgot to specify a zone name on the command line). Fixes [ticket 800](#).
- [commit 315dd2e](#): Simplify socket listening code, and make sure we always set the nonblocking flag correctly. Patch by Mark Zealey, closes [ticket 664](#).
- [commit b35da1b](#): if_ether.h is in netinet/ not net/ on OpenBSD, thanks Florian Obser.
- [commit 71301b6](#): Replicate gsql backend feature of having separate -auth queries for DNSSEC into oraclebackend. Also lets you disable dnssec if you are not ready for it. Closes [ticket 527](#), patch by Aki Tuomi.
- [commit 2125dac](#): drop unused ignore-rd-bit flag
- [commit 8c1a6d6](#): NSECx optimizations, thanks Kees Monshouwer.
- [commit 664716a](#): drop unused variables in lua backend ([ticket 653](#))
- [commit d8ec70f](#): fix db2 backend includes ([ticket 653](#))
- [commit 6477102](#): add goracle schema, thanks Aki Tuomi.
- [commit 9118638](#): make goraclebackend “at least work”, closes [ticket 729](#), thanks Aki Tuomi.
- [commit e0ad7bb](#): add DS digest type 4 to show-zone output; add algorithm names. Based on a patch by Aki Tuomi, closes [ticket 744](#)
- [commit 61a7fac](#): enable AM_SILENT_RULES, closing [ticket 647](#)
- [commit 837f4b4](#): do a better job at escaping TXT, fixes [ticket 795](#)
- [commit 6ca3fa7](#): add SOA-EDIT INCEPTION-INCREMENT mode, thanks stbuehler
- [commit 6159c49](#): Add connection info to sql-connect message
- [commit 9f62e34](#), [commit 0fc965f](#), [commit 2035112](#): Added EUI48 and EUI64 record types
- [commit f9cf6d9](#): cut the number of database queries in half for AXFR-in, thanks Kees Monshouwer.
- [commit c87f987](#): add default for SOA contact e-mail
- [commit bb4a573](#): move random backend to modules, thanks Kees Monshouwer.
- [commit 1071abd](#): restyle builtin webserver page, thanks Christian Hofstaedtler.
- [commit cd5e158](#): correct bogus use of poll(2) related constants, improving non-Linux portability. Thanks Wouter de Jong.
- [commit 27ff60a](#): make sure our NSEC(3)s for names with spaces in them are correct. Reported by Jimmy Bergman. Includes test.
- [commit 116e28a](#): reduce log level of successful gpgsql/gsqlite3 connection to Info
- [commit b23b90a](#): Metadata update is now in the same transaction as the AXFR. This improves slaving speed tremendously, especially for SQLite users. Patch by Kees Monshouwer.
- [commit 4620e8a](#): Added zone2json, thanks Aki Tuomi.
- [commit f0fa8b6](#): Fix remotebackend setdomainmetadata return value handling. Fix by Aki Tuomi, closes [ticket 740](#).
- [commit 80e82d6](#): log control listener abort even more explicitly.
- [commit 7c0cb15](#), [a718d74](#): support automake 1.12

- [commit 3fe22eb, 6707cb1](#): update autoconf/automake preamble to non-deprecated variant, thanks Morten Stevens
- [commit 6c4e531](#): disarm dead code that causes gcc crashes on ARM, thanks Morten Stevens.
- [commit 36855b5](#): if we failed to make a new UDP socket, we'd report a confusing error about it.
- [commit 1b8e5e6](#): autoconf support for oracle, thanks Aki Tuomi. Closes [ticket 726](#).
- [commit 8ac0c06](#): allow setting of some oracle env vars. Patch by Aki Tuomi, closes [ticket 725](#).
- [commit 45e845b](#): add example.rb sample script for remotebackend, thanks Aki Tuomi.
- [commit 950bddd](#): add pdnssec generate-zone-key command, thanks Aki. Closes [ticket 711](#).
- [commit 2c03cde](#): Replace select with waitForData in remotebackend. Patch by Aki Tuomi, closes [ticket 715](#).
- [commit 450292c](#): accept ANY responses during recursive forwarding, thanks Jan-Piet Mens.
- [commit d9dd76b](#): actually clean up unix domain sockets too after use.
- [commit 36758d2](#): merge [ticket 476](#) by Aki Tuomi, providing default-ksk/zsk-algorithms/size configuration parameters for pdnssec.
- [commit 2f2b014](#): apply variant of code in [ticket 714](#) so we can launch pipe backend scripts with parameters, plus add experimental code that if pipe-command is a unix domain socket, we use that.
- [commit 9566683](#): merge patch from ticket 712 addressing memory leak in remotebackend, thanks Aki.
- [commit fb6ed6f](#): explicitly set domain id during bindbackend superslave domain create, thanks Kees Monshouwer&Aki Tuomi.
- [commit 69bae20](#): use private temp dir when running under systemd, thanks Morten Stevens&Ruben Kerkhof.
- [commit b26a48a](#): fix rapidjson usage in remotebackend, patch by Aki Tuomi. Closes [ticket 697](#).
- [commit da8e6ae](#): also answer questions with : in them.
- [commit ef1c4bf](#): also spot trailing dots on CNAME content, thanks Jan-Piet Mens and Ruben d'Arco.
- [commit fb31631](#): only setCloseOnExec on valid sockets

19.3.15 PowerDNS Authoritative Server 3.2

Released January 17th, 2013

This is a stability and conformity update to 3.1. It mostly makes our DNSSEC implementation more robust, and improves interoperability with various validators. 3.2 has received very extensive testing on a lot of edge cases, verifying output both against common validators and compared against other authoritative servers.

Warning: Version 3.2 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 or 3.1. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Downloads

- [Official download page](#)
- [native RHEL5/6 packages from Kees Monshouwer](#)
- [additional third-party builds](#)

In addition to all the changes below, we now auto-build semi-static packages. Relevant changes to make that possible are in [commit 2849](#), [commit 2853](#), [commit 2858](#), [commit 2859](#), [commit 2860](#).

Changes between 3.2-RC4 and the final 3.2 release

- Aki Tuomi contributed a bunch of fixes to our crypto drivers. Code in [commit 3036](#) and [commit 3055/commit 3057](#).
- The ksklzsksk argument for pdnssec import-zone-key was required while it should be optional. Fixed in [commit 3051](#).

Changes between 3.2-RC3 and 3.2-RC4

- The experimental undocumented bindbackend superslave mode would break the first added domain until a restart. Fixed by Kees Monshouwer in [commit 3013](#).
- Sander Hoentjen reported an issue with our choice of ports for outgoing TCP connections. Investigating it turned up that we were randomizing TCP connections on purpose while leaving UDP port choice to the kernel, which should be the other way around. Fixed in [commit 3014](#), closing [ticket 643](#) and [ticket 644](#).
- Aki Tuomi contributed some autoconf code to use mysql_config if it is available. Code in [commit 3015](#) and [commit 3019](#), closing [ticket 458](#).
- The MongoDB backend was removed at the author's request, as it does not work with any current libmongo versions. Change in [commit 3017](#).
- Mark Zealey discovered we were retrieving the ascii powerdns version string for each packet, not just for version string queries. Fixed in [commit 3018](#), closing [ticket 651](#).
- Our new json code would not compile on solaris 9 and 10 due to lack of strcasestr. Juraj Lutter contributed a portable version in [commit 3020](#).
- Mark Zealey noted that RRs with low TTLs could lower our query-cache-ttl persistently. Fixed in [commit 3023](#), closing [ticket 662](#).
- pdnssec now honours module-dir, patch by Fredrik Danerklint in [commit 3026](#).

Changes between 3.2-RC2 and 3.2-RC3

- Michael Scheffler noticed that the lazy-recursion setting had no effect at all. Setting removed in [commit 3003](#).
- Mark Zealey found that an earlier performance improvement could cause crashes under high load, with lots of IPs configured in local-address and receiver-threads higher than 1. Fixed in [commit 3005](#).

Changes between 3.2-RC1 and 3.2-RC2

- The udp-queries metric would only count on the first thread launched, instead of on all threads. Additionally, it was initialised at MAXINT at startup, instead of at 0. Both issues fixed by Kees Monshouwer in [commit 2999](#), closing [ticket 491](#) and [ticket 582](#).
- Aki Tuomi contributed zone2json, a great way for programmers to benefit from our zone file parser. Code in [commit 2997](#), closes [ticket 509](#).
- Our DNS TXT parser is not 8-bit safe, but our DNS TXT writer assumes the reader is! Reported by Jan-Piet Mens in [ticket 541](#), [commit 2993](#) fixes our writer but not yet our parser.
- Ruben d'Arco did some improvements to the MyDNS backend, and provided a full test suite for it, that we now run after every commit. Code in [commit 2988](#).
- Some exceptions from backends would lose their meaning while bubbling up. Fixed by Aki Tuomi in [commit 2985](#), closing [ticket 639](#).
- The packet-cache honours max reply length while matching cached packets against queries, but not EDNS status. This would mean that EDNS-enabled replies with a 512 reply len could be returned on non-EDNS

queries. Spotted while investigating a report from Winfried Angele, patched by Ruben d'Arco in [commit 2982](#), closing [ticket 630](#).

- Errors involving creating, deletion or changing permissions on the control socket were unclear. Ruben d'Arco improved this in [commit 2981](#).
- pipe-timeout was always documented to be in milliseconds, but it turns out it was in seconds! [commit 2971](#) changes them to actually be in ms, and 'increases' the default from 1000 seconds to 2000 milliseconds.
- Some exceptions would get dropped during inbound AXFR, yielding a log file that says 'transaction started' and nothing after that, making AXFR fail silently. [commit 2976](#) and [commit 2977](#) improve this somewhat.
- We now error out on empty labels inside of names (www.example.com) instead of generating bogus reply packets. Code in [commit 2972](#), reported by several users.
- Doing chmod before chown, instead of the other way around, apparently avoids requiring a whole SELinux capability. Reported by Sander Hoentjen, fixed in [commit 2965](#).
- Christian Hofstaedtler fixed a bug in our Debian init.d script. Code in [commit 2963](#).
- Superslave errors ('Unable to find backend willing to host ..') now include the NSset found at the master, to aid debugging. Code in [commit 2887](#).
- [commit 2874](#) in RC1 broke compilation without SQLite3 and made query logging unreliable. Fixed in [commit 2888](#), [commit 2889](#).
- The dnsreplay tool now processes single packet pcaps. Fix in [commit 2895](#).
- PowerDNS always derives NSEC/NSEC3 from the actual zone content. To accommodate this, zone2sql now drops NSEC/NSEC3 records, as those should never be in a PowerDNS backend directly ([commit 2915](#)), bindbackend ignores NSEC/NSEC3 while reading zonefiles ([commit 2917](#)) and pdnssec reports NSEC/NSEC3 in the database as an error condition ([commit 2918](#)).
- The bindbackend now ignores NSEC/NSEC3 records while reading zonefiles. Change in [commit 2917](#).
- An EXPERIMENTAL feature ('direct-dnskey') for reading ZSKs from the records table/your BIND zone-file was added in [commit 2920](#), [commit 2921](#), [commit 2922](#).
- While fully optional, PowerDNS supports direct RRSIG queries. Kees Monshouwer improved on our behaviour for those queries in [commit 2927](#).
- IPv6 glue situations require AAAA records for the receiving end of a delegation in the ADDITIONAL section of a referral. This was supported ('do-ipv6-additional-processing') but not enabled by default. [commit 2929](#) enables it by default.
- pdnssec check-zone now warns for CNAME-and-other data at names in your zones. Code by Ruben d'Arco in [commit 2930](#).
- Positive ANY-responses would include a spurious NSEC3. Corrected in [commit 2932](#) and [commit 2933](#), cleaned up by Kees Monshouwer in [commit 2935](#).
- The ldapbackend now allows overriding the base dn for AXFR subtree search. Fixed in [commit 2934](#), closing [ticket 536](#).

Changes below are in 3.2-RC1 and up.

DNSSEC changes in 3.2

- Kees Monshouwer did a tremendous amount of work to improve and perfect our DNSSEC implementation, mostly in the NSEC3 area. Code in [commit 2687](#), [commit 2689](#), [commit 2691](#), fixing [ticket 486](#), [ticket 537](#), [ticket 540](#). He also implemented support for Empty Non-Terminals, code in [commit 2721](#), [commit 2732](#), [commit 2745](#), fixing [ticket 127](#) and [ticket 558](#).
- Presigned wildcard operation was improved with the help of many parties (see commit message for [commit 2676](#)). Presigned operation was also changed to be more consistent with master/live-signing operation. Code and a full test suite in [commit 2709](#), which also improves TTL behaviour for various situations. Fixes [ticket 460](#), [ticket 533](#), [ticket 559](#).

- Depending on database & locale settings, names starting with underscore would sometimes cause broken records. [commit 2710](#) contains schema and code changes for the gpgsql and gmysql backends to sort this (no pun intended) definitively, closing [ticket 550](#). In addition, a `pdnssec test-schema` command was added (experimental and incomplete). It can be used to verify underscore sorting and a few other parameters of the database. Code in [commit 2714](#).
- We now always include an EDNS section in responses to queries that also had an EDNS section. This was thought to improve BIND interoperability, but this turned out to be false. In any case, this change improves standards compliance. Spotted by Mats Dufberg, code in [commit 2649](#).
- It turns out we were storing Botan keys the wrong way. Botan did not care but Polar did, causing interoperability problems. Fixed in [commit 2720](#), with the kind help of Paul Bakker of PolarSSL. Fixes [ticket 492](#) as reported by Florian Obser via Debian.
- `pdnssec add-zone-key` now defaults to RSASHA256, like `secure-zone` already did. Code in [commit 2692](#).
- `pdns_control purge` now also purges DNSSEC-related caches (keys and metadata). Code in [commit 2694](#), by Ruben d'Arco. Fixes [ticket 530](#).
- The signer thread would die in specific situations, leaving you with a non-working but very busy system. Fixed in [commit 2668](#), [commit 2670](#), closing [ticket 517](#).
- `pdnssec secure-zone` now warns when you just signed a slave zone. Suggested by Mark Scholten, code in [commit 2795](#), closes [ticket 592](#).
- `pdnssec check-zone` now warns about out-of-zone data. Patch by Kees Monshouwer in [commit 2826](#), closing [ticket 604](#).
- `pdnssec` now honours `^no-config`. Patch by Kees Monshouwer in [commit 2810](#).
- Various fixes for `bindbackend` presigned operation, mostly by Kees Monshouwer. Code in [commit 2815](#), closing [ticket 600](#).
- `Bindbackend` could get confused about domain metadata, sometimes even causing hangs. Fixes by Kees Monshouwer in [commit 2819](#) and [commit 2834](#), closing [ticket 600](#) and [ticket 603](#).
- SQL queries in `gsql` backends that reference the `domain_id` column have been made explicit about from what table they want this column. This makes it easier to operate custom schemas without changing the queries. Fix by Nicky Gerritsen in [commit 2821](#).
- In various situations involving CNAMEs and wildcards, and for ANY queries involving CNAMEs, we would sometimes return bogus results. Fixed in [commit 2825](#) by Kees Monshouwer.
- `rectify-zone` accidentally set `auth=1` on NS records of secure delegations. Reported by George Notaras, fixed by Kees Monshouwer in [commit 2831](#), closing [ticket 605](#).
- The DNSSEC signature cache now actually gets cleaned up, avoiding lasting spikes in memory usage every thursday. Code in [commit 2836](#) and [commit 2843](#), closing [ticket 594](#).
- Signatures used to roll at midnight on thursday. We now roll them one hour after midnight, with inception still set to midnight, to allow for some variations in clock quality on resolvers. Code in [commit 2857](#).
- Duplicate records (same name/type/content/priority) would sometimes get broken RRSIGs during outgoing AXFR. Fixed in [commit 2856](#).
- A root zone (name=""") with DNSSEC would cause crashes in some situations. Reported by Luuk Hendriks. Fixed in [commit 2867](#), [commit 2868](#), closing [ticket 614](#).
- Direct RRSIG queries for zones with auto-completed SOA records would cause trouble. Reported by Kees Monshouwer and fixed by him in [commit 2869](#).
- When a name is matched only by a wildcard, but the type in the query is not present, we would be lacking one NSEC(3) record to prove the existence of the wildcard. Fixed by Kees Monshouwer in [commit 2872](#) and [commit 2873](#).
- Luuk Hendriks spotted that our PolarSSL RSA key generation code was using inferior entropy. This can be important on virtual machines with badly implemented clocks. Fixed in [commit 2876](#), closing [ticket 615](#).

Non-DNSSEC improvements/changes

- Bindbackend would sometimes crash on startup, due to a `sync_with_stdio` call. This call has been moved to `pdns_server` proper to occur before any threads are spawned, avoiding race conditions in this call. Note that this crash has only been observed twice in thousands of regression test runs and has never been reported in the real world. Change in [commit 2882](#).
- Leen Besselink submitted query logging support for the SQLite3 parts in the bindbackend. Code in [commit 2874](#).
- Multi-backend operation would sometimes cause garbage domain IDs to be passed to backends. Reported by Kees Monshouwer and fixed by him in [commit 2871](#).
- Bindbackend would sometimes crash during reloads/rediscovers. The changes in [commit 2837](#) get rid of the crash, at the cost of returning SERVFAIL during reloads. Closes [ticket 564](#).
- Our label decompression code was naive, causing troubles for slaving of very specifically formatted zones. Fix in [ticket 2822](#), closes [ticket 599](#).
- Bindbackend slaves would choke on unknown RR types and do silly things with RP and SRV records. Fixed in [commit 2811](#) and [commit 2812](#).
- The luabackend can now compile against Lua 5.2. Patch by Fredrik Danerklint in [commit 2794](#), additional luabackend compile fixes in [commit 2854](#).
- A new backend, the ‘Remote backend’ [Remote Backend](#) was submitted by Aki Tuomi. It aims to replace the pipebackend with a better protocol and support for more connection methods, including HTTP. Code in [commit 2755](#), [commit 2756](#), [commit 2757](#), [commit 2758](#), [commit 2759](#), [commit 2824](#), closing [ticket 529](#), [ticket 597](#).
- The `gsqLite` (SQLite 2) backend was removed. We were not aware of any users and it was not actually working anyway. Changes in commits [2773-2777](#), closing [ticket 565](#).
- Various `tinydnsbackend` improvements: `ignore-bogus-records` option; TAI offset updated; strip dots on names where suitable; various internal improvements. Code in [commit 2762](#).
- `gpgsql` no longer logs the database password in connection errors. Code in [commit 2609](#), [commit 2612](#), closing [ticket 459](#).
- You can now finally specify `0.0.0.0` or `::` as local-address/local-ipv6 without getting replies from the wrong address. This much-requested feature is implemented in [commit 2763](#), [commit 2766](#), [commit 2779](#) and [commit 2781](#). Tested on Linux, FreeBSD and Mac OS X.
- 3.2 can be reliably built with or without Lua. This and many other configure/compile-related fixes in [commit 2610](#), [commit 2611](#) / [ticket 461](#), [commit 2666](#), [commit 2671](#), [commit 2672](#) / [ticket 522](#), [commit 2673](#) / [ticket 522](#), [commit 2696](#) / [ticket 555](#), [commit 2697](#) / [ticket 457](#), [commit 2698](#), [commit 2708](#), [commit 2742](#) / [ticket 462](#), [commit 2752](#) / [ticket 437](#), [commit 2764](#), [commit 2809](#), [commit 2844](#), [commit 2845](#), [commit 2846](#), [commit 2881](#).
- Juraj Lutter contributed AXFR-SOURCE per zone metadata settings. Code in [commit 2616](#).
- `initscripts` now have exit codes, submitted by Sander Hoentjen. Code in [commit 2728](#). `Guardian` now returns 0 instead of 1 when receiving SIGTERM, requested by Morten Stevens of Fedora. Code in [commit 2717](#).
- Mark Zealey submitted various performance improvement patches and suggestions. Accepted as [commit 2729](#) / [ticket 579](#), [commit 2730](#) / [ticket 584](#), [commit 2731](#) / [ticket 583](#), [commit 2768](#) / [ticket 578](#)). Please see commit messages for more details.
- `pdnssec` `check-all-zones` now reuses database connections, avoiding a socket exhaustion issue in some situations. Code in [commit 2749](#), closes [ticket 519](#).
- Ruben d’Arco submitted various improvements regarding trailing dots. Additional lookups now try harder, `pdnssec` errors about trailing dots in names, `pdnssec` warns about trailing dots in names inside content fields, AXFR now strips the dot from SRV hostnames. Code in [commit 2748](#), fixes [ticket 289](#).
- Pre-3.0, backends would get cycled if they threw the right error. 3.2 reinstates this behaviour, as it is more robust. Change in [commit 2734](#) (reverting [commit 2100](#)), fixes [ticket 386](#).

- PowerDNS auth does not use the `select()` kernel/library call anymore. This means fd-numbers over 1023 (and, in general, more than 1024 sockets, including more than 1024 listening sockets) should now work reliably. Code in [commit 2739](#), [commit 2740](#), fixes [ticket 408](#).
- `gmysql` users can now specify the 'group' we connect as, using the `gmysql-group` setting. Submitted by Kees Monshouwer, code in [commit 2770](#), [commit 2771](#), [commit 2778](#), [commit 2780](#), closing [ticket 463](#).
- The Linux-only traceback handler is now optional (use `traceback-handler=off` to disable it). Suggested by Marc Haber. Change in [commit 2798](#), closes [ticket 497](#).
- We now use `IPV6_V6ONLY` to bind IPv6 sockets. This ensures consistent behaviour between different operating systems. Change in [commit 2799](#).
- MySQL connections are now logged at a higher loglevel, reducing log clutter. Change in [commit 2800](#).
- We now ship a `systemd` unit file in `contrib/`. Added in [commit 2847](#) and [commit 2848](#), submitted by Morten Stevens.

Assorted bugfixes

- If a slave domain is removed while a transfer for it is queued, we no longer try the transfer. This also avoids a rare crash in similar circumstances. Code in [commit 2802](#), closes [ticket 596](#).
- When using `pdnssec` with `gsql` backends, sometimes an `SSqlException` would pop up without any useful information. This no longer happens and errors are now in general more meaningful. Fix in [commit 2803](#).
- `zone2sql` now uses correct string syntax for PostgreSQL. This is needed for importing with the changed default settings in PostgreSQL 9.2 and up. Code in [commit 2797](#), closes [ticket 471](#).
- We no longer send v6 notifications if v6 is not available. Same for IPv4. Code in [commit 2772](#), fixes [ticket 515](#).
- We would sometimes serve stale data after an incoming AXFR. Reported by Martin Draschl, fixed by Ruben d'Arco in [commit 2699](#), closing [ticket 525](#).
- Duplicate incoming NOTIFYs could cause PowerDNS to try to insert the same domain name into a database twice. Fixed in [commit 2703](#), closing [ticket 453](#).
- `pdnssec show-zone` now works on a zone that has any number of keys, instead of requiring active keys. Reported by Jeroen Tushuizen of myH2Oservers, code in [commit 2769](#), closes [ticket 586](#).
- `pdns-control notify-host` now accepts v6 literals. Reported by Christof Meerwald, fixed in [commit 2704](#).
- The `tinydnsbackend` no longer chokes on questions longer than 64 bytes. Code in [commit 2622](#).
- `*-all-domains` commands in `pdnssec` now work with Postgres (`pgpsql`) too. Code in [commit 2645](#), closing [ticket 472](#).
- We would sometimes leave the opcode of an outgoing packet uninitialized. Fixed in [commit 2680](#), closing [ticket 532](#).
- `nproxy` can now listen on a configurable port. Code in [commit 2684](#), fixes [ticket 534](#).
- Improve `mydnsbackend` for SOA queries. Code in [commit 2751](#), fixes [ticket 439](#), by Ruben d'Arco.
- Various non-functional fixes that make Valgrind happy (note that Valgrind was right to complain in all of these situations), in [commit 2715](#), [commit 2716](#), [commit 2718](#).

19.3.16 PowerDNS Authoritative Server 3.1

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

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 the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

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!

Downloads

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

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 bind backend 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](#).

19.3.17 Changes between RC1 and RC2

- We imported the TinyDNS backend by Ruben d’Arco. Code mostly in [commit 2559](#). See [TinyDNS Backend](#).
- 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 dnsmog 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 presigned. 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](#).
- 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 gpysql 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”

19.3.18 Authoritative Server version 2.9.22.6

Warning: The 2.9.22.x series of releases is end-of-life and unsupported. It contains many issues and potential security problems. We urge you to upgrade to a recent version of PowerDNS!

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.

19.3.19 Authoritative Server version 2.9.22.5

Warning: The 2.9.22.x series of releases is end-of-life and unsupported. It contains many issues and potential security problems. We urge you to upgrade to a recent version of PowerDNS!

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 [PowerDNS Security Advisory 2012-01](#).

- 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](#))

19.3.20 PowerDNS Authoritative Server 3.0.1

Warning: The DNSSEC implementation of PowerDNS Authoritative Server 3.0 and 3.0.1 contains many issues regarding CNAMEs, wildcards and (in)secure delegations. If you use any of these, and you use DNSSEC you MUST upgrade to 3.1 or beyond!

3.0.1 consists of 3.0, plus the patch from [PowerDNS Security Advisory 2012-01](#)

19.3.21 PowerDNS Authoritative Server 3.0

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

Warning: Version 3.0 of the PowerDNS Authoritative Server is a major upgrade if you are coming from 2.9.x. Please refer to the [Upgrade documentation](#) for important information on correct and stable operation, as well as notes on performance and memory use.

Warning: The DNSSEC implementation of PowerDNS Authoritative Server 3.0 and 3.0.1 contains many issues regarding CNAMEs, wildcards and (in)secure delegations. If you use any of these, and you use DNSSEC you MUST upgrade to 3.1 or beyond!

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 [Serving authoritative DNSSEC data](#). 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](#)
-

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, Vasily G Tolstov, Brielle Bruns, Evan Hunt, Ralf van der Enden, Marc Laros, Serge Belyshev, Christian Hofstaedtler, Charlie Smurthwaite, Nikolaos Milas, ..

Known issues as of RC3

- Not all new features are fully documented yet

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](#)).
- 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](#).

Other major new features

- 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-notify’ 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 MBOFW 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](#).

19.3.22 Authoritative Server version 2.9.22

Warning: The 2.9.22.x series of releases is end-of-life and unsupported. It contains many issues and potential security problems. We urge you to upgrade to a recent version of PowerDNS!

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 ccounts** 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 “[Notification proxy \(nproxy\)](#)”. 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 rcvfrom 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](#).

19.3.23 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 [PowerDNS Security Advisory 2008-02](#).

19.3.24 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 [PowerDNS Security Advisory 2008-02](#).

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.

19.3.25 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 [Slave operation](#). 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 AFSDB 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](#).

19.3.26 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](#).

19.3.27 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 [PowerDNS Recursor](#).
- 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. [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](#).

19.3.28 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 [Tools to analyse DNS traffic](#) 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 [PipeBackend protocol](#) 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 [Recursion](#) 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 `writv`, 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 [Tools to analyse DNS traffic](#) for more details.
- Generic Oracle Backend, sponsored by Register.COM. See [Oracle specifics](#).

19.3.29 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_recurser 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_recurser ^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_recurser 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.

19.3.30 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

19.3.31 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 example.org | 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

19.3.32 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 [Recursion](#).

19.3.33 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_recurser` 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_recurser` 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_recurser` now survives the expiration of all its root records, most often caused by prolonged disconnection from the net.

19.3.34 Version 2.9.12

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

New features

- `^^version` command (requested by Mike Benoit)
- delegation-only, a Verisign special.
- Generic [SQLite](#) support, by Michel 'Who da man?' Stol. See [Generic SQLite backend](#).
- `init.d` script for `pdns_recurser`
- 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_recurser`, `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_recurser` 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.

19.3.35 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_recurser 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.

19.3.36 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_recurser 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)

19.3.37 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!

19.3.38 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.

19.3.39 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 [LDAP Backend](#).
- 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.

19.3.40 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 PowerDNS is also almost seamless. As a result, ‘non-lazy recursion’ has been dropped. See [Recursion](#) 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 PowerDNS 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 syncrec 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.

19.3.41 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 [PowerDNS Recursor](#) 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 [Security](#). 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 [Generic SQL backends](#).
- A recursing nameserver! See [PowerDNS Recursor](#).
- 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).

19.3.42 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 `example.org` 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 'example.org' to 195.193.163.3
pdns[17495]: Queued notification of domain 'example.org' to 213.156.2.1
pdns[17520]: AXFR of domain 'example.org' initiated by 195.193.163.3
pdns[17520]: AXFR of domain 'example.org' to 195.193.163.3 finished
```

(continues on next page)

(continued from previous page)

```
pdns[17521]: AXFR of domain 'example.org' initiated by 213.156.2.1
pdns[17521]: AXFR of domain 'example.org' to 213.156.2.1 finished
pdns[17495]: Removed from notification list: 'example.org' to 195.193.163.3 (was_
↳acknowledged)
pdns[17495]: Removed from notification list: 'example.org' 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 example.org
Added to queue
pdns[17492]: Notification request for domain 'example.org' received
pdns[17492]: Queued notification of domain 'example.org' to 195.193.163.3
pdns[17492]: Queued notification of domain 'example.org' to 213.156.2.1
pdns[17495]: Removed from notification list: 'example.org' to 195.193.163.3 (was_
↳acknowledged)
pdns[17495]: Removed from notification list: 'example.org' 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 “[pdns_control](#)” and “[pdns_control commands](#)”.

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 [Pdns_control commands](#). 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.

19.3.43 Version 2.9.2

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

Warning: The Generic MySQL backend, part of the Generic MySQL & PostgreSQL backend, is now the DEFAULT! The previous default, the ‘mysql’ backend (note the lack of ‘g’) is now DEPRECATED. This was the source of much confusion. The ‘mysql’ backend does not support MASTER or SLAVE operation. The Generic backends do.

To get back the mysql backend, add `^with-modules="mysql"` or `^with-dynmodules="mysql"` if you prefer to load your modules at runtime.

Bugs fixed

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

Documentation

- Documented the 'random backend', see [Random Backend](#).
- Wichert Akkerman contributed three manpages.
- Building PowerDNS on Unix is now documented somewhat more, see [Compiling PowerDNS on Unix](#).

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 [XDB Backend](#). 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.

19.3.44 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).

19.3.45 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.

19.3.46 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.

19.3.47 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 PowerDNS 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 (192.0.2.0/24) 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.

19.3.48 Version 2.6.1

Quick fix release for a big cache problem.

19.3.49 Version 2.6

Performance release. A lot of work has been done to raise PowerDNS performance to staggering levels in order to take part in benchmarking efforts. Together with our as yet unnamed partner, PowerDNS 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 PowerDNS itself quicker but also by lowering the number of backend queries typically needed. Operators will typically see PowerDNS 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 [Backend writers' guide](#).

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'.

19.3.50 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 [Backend writers' guide](#).

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.

19.3.51 Version 2.5

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

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

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

New features

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

PowerDNS 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 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 [Authoritative Server Performance](#) 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 PowerDNS 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 packet-cache, 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.

19.3.52 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 **Backend writers’ guide**.

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.
- Webserver now displays peak queries/second 1 minute average.

Bugs fixed

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

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

19.3.53 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 *Backend writers’ guide*

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 *Generic MySQL backend*.

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.

19.3.54 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 [Backend writers’ guide](#)

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 PowerDNS differed from BIND in authoritative capabilities.

If you were seeing signal 6 errors in PowerDNS 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 PowerDNS 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 [pdns_control](#) 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 21](#), [*Index of all Authoritative Server settings*](#).

19.3.55 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 [Backend writers' guide](#).

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

Bugs fixed

- RPM releases sometimes overwrote previous configuration files. Thanks to Jorn Ekkelenkamp of Hubris/ISP Services for reporting this.
- TCP recursion sent out overly large responses due to a 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-url** in the configuration.
- Unlike other nameservers, local data did not override the internet for recursing queries. This has mostly been brought into conformance with user expectations. If a recursive question can be answered entirely from local data, it is. To restore old behaviour, disable **lazy-recursion**. Also see [Recursion](#).

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 [Oracle backend](#).
- 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 [Zone2sql](#).
- **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 [Authoritative Server Performance](#) and [pdns_control](#).

19.3.56 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.

19.3.57 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 PowerDNS to reload. Present in all earlier 2.0 versions.

19.3.58 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.

19.3.59 Version 2.0 Release Candidate 1

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

Developers: this version is compatible with 1.99.11 backends.

- Implemented native packet response parsing code, allowing Windows to perform AXFR and NS and SOA queries.
- This is the first version for which we have added support for Darwin 6.0, which is part of the forthcoming Mac OS X 10.2. Please note that although this version is marked RC1, that we have not done extensive testing yet. Consider this a technology preview.
 - The Darwin version has been developed on Mac OS X 10.2 (6C35). Other versions may or may not work.
 - Currently only the random, bind, mysql and pdns backends are included.
 - The menu based installer script does not work, you will have to edit pathconfig by hand as outlined in chapter 2.
 - On Mac OS X Client, PowerDNS 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.

19.3.60 Version 1.99.12 Prerelease

The Windows release! See [Installing on Microsoft Windows](#). 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 [ODBC backend](#).
- IBM DB2 Universal Database backend available for Linux. See [DB2 backend](#).
- 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 [Virtual hosting](#)), the additional init.d scripts would not function correctly and interface with other pdns instances.
- PowerDNS 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 [pdns_control](#)) 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, *Security settings & considerations*](#)), the local socket again moves back to /var/run.
- pdns_control now has a 'version' command. See [Section 1.1, "pdns_control"](#).

19.3.61 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 PowerDNS. Furthermore, a few bugs have been fixed

- Lines with only whitespace in zone files confused PowerDNS (thanks Henk Wevers)
- PowerDNS did not properly parse TTLs with symbolic suffixes in zone files, ie 2H instead of 7200 (thanks Henk Wevers)

19.3.62 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!

PowerDNS 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

- PowerDNS 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 [Master/Slave operation & replication](#) for lots of details.
- **Zone2sql** in PostgreSQL mode now populates the ‘domains’ table for easy master, slave or native replication support.
- Ability to run on IPv6 transport only
- Logging can now happen under a ‘facility’ so all PowerDNS messages appear in their own file. See [Operational logging using syslog](#).
- Different OS releases of PowerDNS 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.

19.3.63 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

- PowerDNS 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 PowerDNS) 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 - PowerDNS 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 [Backend writers’ guide](#).
- 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 [Supermaster automatic provisioning of slaves](#).
- Recursing backend can now live on a non-standard (!=53) port. See [Recursion](#).
- 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 [Backend writers’ guide](#).
- Added Bindbackend to internal regression tests, confirming that it is compliant.

19.3.64 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 PowerDNS 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 [Recursion](#). 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 [Slave operation](#)
- All record types now documented, see [Supported record types and their storage](#).

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

Missing features

Features present in this document, but disabled or withheld from the current release - gmysqlbackend, oraclebackend

19.3.65 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.

Known bugs

Wildcard CNAMEs do not work as they do with bind.

Missing features

Features present in this document, but disabled or withheld from the current release - gmysqlbackend, oracleback-end

Some of these features will be present in newer releases.

19.3.66 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.

Known bugs

- FreeBSD version does not stop when requested to do so.
- Wildcard CNAMEs do not work as they do with bind.

Missing features

Features present in this document, but disabled or withheld from the current release - gmysqlbackend, oracleback-end

Some of these features will be present in newer releases.

19.3.67 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.

Known bugs

- FreeBSD version does not stop when requested to do so.
- Wildcard CNAMEs do not work as they do with bind.

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.

19.3.68 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.

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.

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.

19.3.69 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, PowerDNS 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. - Webserver does not crash anymore on ringbuffer resize

Feature enhancements

- AXFR can now be disabled, and re-enabled per IP address
- ^help accepts a parameter, will then show only help items with that prefix.
- zone2sql now accepts a ^zone-name parameter
- BindBackend maturing - 9500 zones parsed in 3.5 seconds. No longer case sensitive.

Performance enhancements

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

Known bugs

- FreeBSD version does not stop when requested to do so.
- BindBackend cannot parse zones with \$GENERATE statements.

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.

19.3.70 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.

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	192.0.2.4
	IN	A	192.0.2.5

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.

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.

19.3.71 Version 1.99.1 Early Access Prerelease

This is the first public release of what is going to become PowerDNS 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.

Known bugs

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

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.

END OF LIFE STATEMENTS

The currently supported release train of PowerDNS Authoritative Server is 4.1.

PowerDNS Authoritative Server 4.0 will only receive correctness, stability and security updates.

PowerDNS Authoritative Server 3.x and 2.x are end of life, and will not receive any updates, not even security fixes.

Note: Users with a commercial agreement with PowerDNS.COM BV or Open-Xchange can receive extended support for releases which are End Of Life. If you are such a user, these EOL statements do not apply to you.

20.1 PowerDNS Authoritative Server 3.x

1st of December 2017

The PowerDNS Authoritative Server 3.x releases are no longer supported, and will not receive any further updates, not even for security purposes.

All users are urged to upgrade to version 4.1. To upgrade from 3.x to 4.x, [follow these instructions](#)

If you need help with upgrading, we provide [migration services](#) to our supported users. If you are currently running 3.x and need help to tide you over, we can also provide that as part of a [support agreement](#).

20.2 PowerDNS Authoritative Server 2.x

21st of May 2015 (updated January 2017)

PowerDNS Authoritative Server 2.9.22 was released in January 2009. Because of its immense and durable popularity, some patch releases have been provided, the last one of which (2.9.22.6) was made available in January 2012.

The 2.9.22.x series contains a number of probable and actual violations of the DNS standards. In addition, some behaviours of 2.9.22.x are standards conforming but cause interoperability problems today. Finally, 2.9.22.4 and earlier are impacted by [PowerDNS Security Advisory 2012-01](#), which means PowerDNS can be used in a Denial of Service attack.

Although we have long been telling users that we can no longer support the use of 2.x, and urging upgrading, with this statement we formally declare 2.x end of life.

This means that any 2.x issues will not be addressed. This has been the case for a long time, but with this statement we make it formal.

To upgrade to 3.x, please consult the [instructions on how to upgrade the database](#). To upgrade from 3.x to 4.x, [follow these instructions](#). If you need help with upgrading, we provide [migration services](#) to our supported users. If you are currently running 2.9.22 and need help to tide you over, we can also provide that as part of a [support agreement](#).

But we urge everyone to move on to PowerDNS Authoritative Server 4.1 or later - it is a faster, more standards conforming and more powerful nameserver!

FREQUENTLY ASKED QUESTIONS

This document lists categorized answers and questions with links to the relevant documentation.

21.1 Replication

Please note that not all PowerDNS Server backends support master or slave support, see the [table of backends](#).

21.1.1 My PowerDNS Authoritative Server does not send NOTIFY messages

Don't forget to enable master-support by setting *master* to *yes* in your configuration. In *master mode* PowerDNS Authoritative Server will send NOTIFYs to all nameservers that are listed as NS records in the zone by default.

21.1.2 My PowerDNS Authoritative Server does not start AXFRs

Don't forget to enable slave-support by setting *slave* to *yes* in your configuration. In *slave mode* PowerDNS Authoritative Server listens for NOTIFYs from the master IP for zones that are configured as slave zones. And will also periodically check for SOA serial number changes at the master.

21.1.3 Can PowerDNS Server act as Slave and Master at the same time?

Yes totally, enable both by saying *yes* to *master* and *slave* in your configuration.

21.1.4 How can I limit Zone Transfers (AXFR) per Domain?

With the ALLOW-AXFR-FROM metadata, See [the documentation](#).

21.1.5 I have a working Supermaster/Superslave setup but when I remove Domains from the Master they still remain on the Slave. Am I doing something wrong?

You're not doing anything wrong. This is the perfectly normal and expected behavior because the AXFR (DNS Zonetransfer) Protocol does not provide for zone deletion. You need to remove the zones from the slave manually or via a custom script.

21.2 Operational

21.2.1 The ADDITIONAL is section different than BIND's answer, why?

21.2.2 My server is not answering with a verbose “ADDITIONAL SECTION” that includes A records for the namservers of the domain queried

The PowerDNS Authoritative Server by default does not ‘trust’ other zones in its own database. You may want to add *out-of-zone-additional-processing* to *yes* in your configuration to tell it to do so. If the domain your nameservers are in are known to the backend they will now be included in the additional section.

21.2.3 PowerDNS does not give authoritative answers, how come?

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!

21.2.4 Master or Slave support is not working, PowerDNS is not picking up changes

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 and you entered exactly the same string to the Domains tables without the ending dot.

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

By default, PowerDNS lets the kernel pick the source address. To set an explicit source address, use the *query-local-address* and *query-local-address6* settings.

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

Please don't ignore what PowerDNS says to you. Furthermore, see the documentation for the *local-address* and *local-ipv6* settings, and use it to specify which IP addresses PowerDNS should listen on. If this is a fail-over address, then the *local-address-nonexist-fail* and *local-ipv6-nonexist-fail* settings might interest you.

21.2.7 Linux Netfilter says your conntrack table is full?

That's a common problem with Netfilter Conntracking and DNS Servers, just tune your kernel variable (`/etc/sysctl.conf`) `net.ipv4.netfilter.ip_conntrack_max` up accordingly. Try setting it for a million if you don't mind spending some MB of RAM on it for example.

21.3 Backends

21.3.1 Does PowerDNS support splitting of TXT records (multipart or multiline) with the MySQL backend?

PowerDNS with the *Generic SQL Backends* do NOT support this. Simply make the “content” field in your database the appropriate size for the records you require.

21.3.2 I see this a lot of “Failed to execute mysql_query” or similar log-entries

Check your MySQL timeout, it may be set too low. This can be changed in the `my.cnf` file.

21.3.3 Which backend should I use? There are so many!

If you have no external constraints, the *Generic MySQL backend*, *Generic PostgreSQL backend* and *Generic SQLite 3 backend* ones are probably the most used and complete.

The Oracle backend also has happy users, the BIND backend is pretty capable too in fact, but many prefer a relational database.

21.3.4 Can I launch multiple backends simultaneously?

You can. This might for example be useful to keep an existing BIND configuration around but to store new zones in, say MySQL. The syntax to use is `launch=bind,gmysql`. Do note that multi-backend behaviour is not specified and might change between versions. This is especially true when DNSSEC is involved.

21.3.5 I’ve added extra fields to the domains and/or records table. Will this eventually affect the resolution process in any way?

No, the *Generic SQL Backends* use several default queries to provide the PowerDNS Server with data and all of those refer to specific field names, so as long as you don’t change any of the predefined field names you are fine.

21.3.6 Can I specify custom sql queries for the gmysql / gpgsql backend or are those hardcoded?

Yes you can override the *default queries*.

BACKEND WRITERS' GUIDE

PowerDNS 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 read [this blog post](#) which has a FAQ and several pictures that help explain what a backend is.

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 PowerDNS 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.

22.1 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.

22.2 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 PowerDNS guarantees that your backend will never be called reentrantly.

Note: Queries for wildcard names should be answered literally, without expansion. So, if a backend gets a question for “*.powerdns.com”, it should only answer with data if there is an actual “*.powerdns.com” name

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 PowerDNS than doing so later on. A real error should be reported by throwing an exception.

PowerDNS 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, where is no need to implement your own `getSOA()`.

Besides direct queries, PowerDNS 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;
```

22.3 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 */
```

(continues on next page)

(continued from previous page)

```
class RandomLoader
{
public:
    RandomLoader()
    {
        BackendMakers().report(new RandomFactory);
        L << Logger::Info << "[randombackend] This is the random backend version " <<
        VERSION " 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 PowerDNS 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 PowerDNS. 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 PowerDNS 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!

22.4 Interface definition

22.4.1 Classes

class DNSResourceRecord

std::string *DNSResourceRecord* : **qname**
Name of this record

QType *DNSResourceRecord* : **qtype**
Query type of this record

std::string *DNSResourceRecord* : **content**
ASCII representation of the right-hand side

uint32_t *DNSResourceRecord* : **ttl**
Time To Live of this record

int *DNSResourceRecord* : **domain_id**
ID of the domain this record belongs to

time_t *DNSResourceRecord* : **last_modified**
If unzero, last time_t this record was changed

bool *DNSResourceRecord* : **auth**
Used for DNSSEC operations. See *Migrating (Signed) Zones to PowerDNS*. It is also useful to check out the `rectifyZone()` in `pdnsutil.cc`.

bool *DNSResourceRecord::disabled*

If set, this record is not to be served to DNS clients. Backends should not make these records available to PowerDNS unless indicated otherwise.

class SOAData

string *SOAData::nameserver*

Name of the master nameserver of this zone

string *SOAData::hostmaster*

Hostmaster of this domain. May contain an @

uint32_t *SOAData::serial*

Serial number of this zone

uint32_t *SOAData::refresh*

How often this zone should be refreshed

uint32_t *SOAData::retry*

How often a failed zone pull should be retried.

u_int32_t *SOAData::expire*

If zone pulls failed for this long, retire records

uint32_t *SOAData::default_ttl*

Difficult

int *SOAData::domain_id*

The ID of the domain within this backend. Must be filled!

DNSBackend **SOAData::db*

Pointer to the backend that feels authoritative for a domain and can act as a slave

22.4.2 Methods

void DNSBackend::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 pkt. If it is, you can retrieve information about who asked the question with the pkt->getRemote() method.

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 PDNSException 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 DNSBackend::list (int domain_id, bool include_disabled = false)

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, PowerDNS will not get confused. If include_disabled is given as true, records that are configured but should not be served to DNS clients must also be made available.

Should return false if the backend does not consider itself authoritative for this zone. Should throw an `PDNSEException` if an error occurred accessing the database. Returning true indicates that data is or should be available.

`bool DNSBackend::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 `PDNSEException` in case a database error occurred.

`bool DNSBackend::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 `PDNSEException` should be thrown.

22.5 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.

22.6 Declaring and reading configuration details

It is highly likely that a backend needs configuration details. On launch, these parameters need to be declared with PowerDNS 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. PowerDNS will call this method after launching the backend.

In the `declareArguments()` method, the function `declare()` is available. The exact definitions:

`void DNSBackend::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 DNSBackend::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", "");
}
```

(continues on next page)

(continued from previous page)

```
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 DNSBackend::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 DNSBackend::mustDo(const string &key)`

Returns true if the variable `key` is set to anything but 'no'.

`const string &DNSBackend::getArg(const string &key)`

Returns the exact value of a parameter.

`int DNSBackend::getArgAsNum(const string &key)`

Returns the numerical value of a parameter. Uses `atoi()` internally

Sample usage from the `BindBackend`: getting the 'check-interval' setting:

```
if(!safeGetBBDomainInfo(i->name, &bbd)) {
    bbd.d_id=domain_id++;
    bbd.setCheckInterval(getArgAsNum("check-interval"));
    bbd.d_lastnotified=0;
    bbd.d_loaded=false;
}
```

22.7 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. PowerDNS 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, PowerDNS 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.

PowerDNS then retrieves the SOA of those domains remotely and locally and creates a list of stale domains. For each of these domains, PowerDNS 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();
}
```

(continues on next page)

(continued from previous page)

```

virtual bool abortTransaction();
virtual bool feedRecord(const DNSResourceRecord &rr, string *ordername=0);
virtual void getUnfreshSlaveInfos(vector<DomainInfo>* domains);
virtual void setFresh(uint32_t id);
    /* ... */
}

```

The mentioned DomainInfo struct looks like this:

class DomainInfo

uint32_t *DomainInfo*::id

ID of this zone within this backend

string *DomainInfo*::master

IP address of the master of this domain, if any

uint32_t *DomainInfo*::serial

Serial number of this zone

uint32_t *DomainInfo*::notified_serial

Last serial number of this zone that slaves have seen

time_t *DomainInfo*::last_check

Last time this zone was checked over at the master for changes

enum *DomainKind* *DomainInfo*::kind

Type of zone

DNSBackend **DomainInfo*::backend

Pointer to the backend that feels authoritative for a domain and can act as a slave

enum **DomainKind**

The kind of domain, one of {Master,Slave,Native}.

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 *DomainInfo*::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 *DomainInfo*::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 *DomainInfo*::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 *DomainInfo*::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 *DomainInfo*::feedRecord(const *DNSResourceRecord* &rr, string *ordername)

Insert this record.

bool *DomainInfo*::commitTransaction()

Make the changes effective. In SQL terms, execute **COMMIT**.

bool *DomainInfo*::abortTransaction()

Abort changes. In SQL terms, execute **ABORT**.

bool *DomainInfo*::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().

PowerDNS 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 PowerDNS is currently:

```
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;
```

22.8 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>&nsset, 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 the *Supermaster and Superslave* documentation.

22.9 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, PowerDNS queries backends for domains that may have changed, and sends out notifications for slave nameservers.

In order to do so, PowerDNS 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 DNSBackend::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 DNSBackend::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

22.10 DNS update support

To make your backend DNS update compatible, it needs to implement a number of new functions and functions already used for slave-operation. The new functions are not DNS update specific and might be used for other update/remove functionality at a later stage.

```

class DNSBackend {
public:
    /* ... */
    virtual bool startTransaction(const string &qname, int id);
    virtual bool commitTransaction();
    virtual bool abortTransaction();
    virtual bool feedRecord(const DNSResourceRecord &rr, string *ordername);
    virtual bool replaceRRSet(uint32_t domain_id, const string& qname, const QType& qtype, const vector<DNSResourceRecord>& rrset);
    virtual bool listSubZone(const string &zone, int domain_id);
    /* ... */
}

```

virtual bool DNSBackend::startTransaction (const string &qname, int id)

See above. Please note that this function now receives a negative number (-1), which indicates that the current zone data should NOT be deleted.

virtual bool DNSBackend::commitTransaction ()

See [above](#).

virtual bool DNSBackend::abortTransaction ()

See `cpp:func:above <DNSBackend::abortTransaction>`. Method is called when an exception is received.

virtual bool DNSBackend::feedRecord (const *DNSResourceRecord* &rr, string *ordername)

See [above](#). Please keep in mind that the zone is not empty because `startTransaction()` was called different.

virtual bool DNSBackend::listSubZone (const string &name, int domain_id)

This method is needed for rectification of a zone after NS-records have been added. For DNSSEC, we need to know which records are below the currently added record. `listSubZone()` is used like `list()` which means PowerDNS will call `get()` after this method. The default SQL query looks something like this:

```
// First %s is 'sub.zone.com', second %s is '*.sub.zone.com'
select content,ttl,prio,type,domain_id,name from records where (name='%s' OR
↳name like '%s') and domain_id=%d
```

The method is not only used when adding records, but also to correct ENT-records in powerdns. Make sure it returns every record in the tree below the given record.

```
virtual bool DNSBackend::replaceRRSet (uint32_t domain_id, const string &qname, const
                                   QType &qt, const vector<DNSResourceRecord>
                                   &rrset)
```

This method should remove all the records with qname of type qt. qt might also be ANY, which means all the records with that qname need to be removed. After removal, the records in rrset must be added to the zone. rrset can be empty in which case the method is used to remove a RRset.

22.11 DNS update support

To make your backend DNS update compatible, it needs to implement a number of new functions and functions already used for slave-operation. The new functions are not DNS update specific and might be used for other update/remove functionality at a later stage.

```
class DNSBackend {
public:
    /* ... */
    virtual bool startTransaction(const string &qname, int id);
    virtual bool commitTransaction();
    virtual bool abortTransaction();
    virtual bool feedRecord(const DNSResourceRecord &rr, string *ordername);
    virtual bool replaceRRSet(uint32_t domain_id, const string& qname, const QType&
↳qt, const vector<DNSResourceRecord>& rrset);
    virtual bool listSubZone(const string &zone, int domain_id);
    /* ... */
}
```

```
virtual bool DNSBackend::startTransaction (const string &qname, int id)
```

See *Read/write slave-capable backends*. Please note that this function now receives a negative number (-1), which indicates that the current zone data should NOT be deleted.

```
virtual bool DNSBackend::commitTransaction ()
```

See *Read/write slave-capable backends*.

```
virtual bool DNSBackend::abortTransaction ()
```

See *Read/write slave-capable backends*. Method is called when an exception is received.

```
virtual bool DNSBackend::feedRecord (const DNSResourceRecord &rr, string *ordername)
```

See *Read/write slave-capable backends*. Please keep in mind that the zone is not empty because `DNSBackend::startTransaction()` was called different.

```
virtual bool DNSBackend::listSubZone (const string &name, int domain_id)
```

This method is needed for rectification of a zone after NS-records have been added. For DNSSEC, we need to know which records are below the currently added record. `listSubZone()` is used like `list()` which means PowerDNS will call `get()` after this method. The default SQL query looks something like this:

```
// First %s is 'sub.zone.com', second %s is '*.sub.zone.com'
select content,ttl,prio,type,domain_id,name from records where (name='%s' OR
↳name like '%s') and domain_id=%d
```

The method is not only used when adding records, but also to correct ENT-records in powerdns. Make sure it returns every record in the tree below the given record.

```
virtual bool DNSBackend::replaceRRSet (uint32_t domain_id, const string &qname, const
                                     QType &qtype, const vector<DNSResourceRecord>
                                     &rrset)
```

This method should remove all the records with qname of type qtype. qtype might also be ANY, which means all the records with that qname need to be removed. After removal, the records in rrset must be added to the zone. rrset can be empty in which case the method is used to remove a RRset.

22.12 Miscellaneous

22.12.1 ENT (Empty Non-Terminal)

You are expected to reply with a DNSResourceRecord having qtype = 0, ttl = 0 and content should be empty string (string length 0)

COMPILING POWERDNS

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:

```
tar xf pdns-VERSION.tar.bz2
cd pdns-VERSION
./configure --with-modules=$MODULES --with-dynmodules=$DYNMODULES
make
make install
```

To compile in modules, specify them as `--with-modules='mod1 mod2 mod3'`, substituting the desired module names. See each *backend specific documentation* for the module names. Each backend has a module name that you look up in this table.

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`.

By default, the *bind*, *mysql* and *random* are compiled into the binary. The *pipe* is, by default, compiled as a runtime loadable module.

23.1 Getting the sources

There are 3 ways of getting the source.

If you want the bleeding edge, you can clone the [repository at GitHub](#) and run `./bootstrap` in the clone.

You can also download snapshot tarballs [here](#).

You can also download releases on the [website](#). These releases are PGP-signed with one of these key-ids:

- `FBAE 0323 821C 7706 A5CA 151B DCF5 13FA 7EED 19F3`
- `1628 90D0 689D D12D D33E 4696 1C5E E990 D2E7 1575`
- `B76C D467 1C09 68BA A87D E61C 5E50 715B F2FF E1A7`
- `16E1 2866 B773 8C73 976A 5743 6FFC 3343 9B0D 04DF`

There is a PGP keyblock with these keys available on <http://powerdns.com/powerdns-keyblock.asc>.

23.2 Dependencies

To build the PowerDNS Authoritative Server, a C++ compiler with support for C++ 2011 is required. This means gcc 4.9 and newer and clang 3.5 and newer. Furthermore, the Makefiles require GNU make, not BSD make.

By default, the PowerDNS Authoritative Server requires the following libraries and headers:

- [Boost](#) 1.35 or newer

- [OpenSSL](#)

To build from git, the following dependencies are also required:

- [ragel](#)
- [bison](#)
- [flex](#)
- [virtualenv](#)

23.3 Optional dependencies

Several options that can be passed to `./configure` can enable and disable different features. These will require additional dependencies

23.3.1 ed25519 support with libsodium

The PowerDNS Authoritative Server can link with [libsodium](#) to support ed25519 (DNSSEC algorithm 15). To detect libsodium, use the `--enable-libsodium` configure option.

23.3.2 ed25519 and ed448 support with libdecaf

[libdecaf](#) is a library that allows the PowerDNS Authoritative Server to support ed25519 and Ed448 (DNSSEC algorithms 15 and 16). To detect libdecaf, use the `--enable-libdecaf` configure option.

23.3.3 systemd notify support

During configure, `configure` will attempt to detect the availability of [systemd](#) or [systemd-daemon](#) headers. To force the use of systemd (and failing configure if the headers do not exist), use `--enable-systemd`. To set the directory where the unit files should be installed, use `--with-systemd=/path/to/unit/dir`.

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.

Starting with 4.0.0, PowerDNS will link in hash and cryptographic primitives from the open source [OpenSSL](#) library.

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 [Sodium](#) library.

24.1 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.

INTERNALS

25.1 How PowerDNS 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, wildcards, referrals and DNSSEC, the algorithm is like this:

When a query for a `qname/qtype` tuple comes in, PowerDNS queries backends to find the closest matching SOA, thus figuring out what backend owns this zone. When the right backend has been found, PowerDNS issues a `qname/ANY` query to the backend. If the response is empty, NXDOMAIN is concluded. If the response is not empty, any contents matching the original `qtype` are added to the list of records to return, and NOERROR is set.

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 PowerDNS backends also contain data. This involves further lookups for A or AAAA records.

After all additional processing has been performed, PowerDNS 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`.

If no SOA was found, a REFUSED is returned.

SUPPORTED RECORD TYPES

This chapter lists all record types PowerDNS 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. Use `pdnsutil check-zone` to validate your zone data.

Note: Whenever the storage format is mentioned, this relates only to the way the record should be stored in one of the *generic SQL* backends. The other backends should use their *native* format.

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

26.1 A

The A record contains an IP address. It is stored as a decimal dotted quad string, for example: ‘203.0.113.210’.

26.2 AAAA

The AAAA record contains an IPv6 address. An example: ‘2001:DB8:2000:bf0::1’.

26.3 AFSDB

A specialised record type for the ‘Andrew Filesystem’. Stored as: ‘#subtype hostname’, where subtype is a number.

26.4 ALIAS

New in version 4.0.0.

The ALIAS pseudo-record type is supported to provide CNAME-like mechanisms on a zone’s apex. See the *howto* for information on how to configure PowerDNS to serve records synthesized from ALIAS records.

26.5 CAA

New in version 4.0.0.

The “Certification Authority Authorization” record, specified in [RFC 6844](#), is used to specify Certificate Authorities that may issue certificates for a domain.

26.6 CERT

Specialised record type for storing certificates, defined in [RFC 2538](#).

26.7 CDNSKEY

New in version 4.0.0.

The CDNSKEY ([Child DNSKEY](#)) type is supported.

26.8 CDS

New in version 4.0.0.

The CDS ([Child DS](#)) type is supported.

26.9 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’.

26.10 DNSKEY

The DNSKEY DNSSEC record type is fully supported, as described in [RFC 4034](#). Enabling DNSSEC for domains can be done with *pdnsutil*.

26.11 DNAME

The DNAME record, as specified in [RFC 6672](#) is supported. However, *dname-processing* has to be set to *yes* for PowerDNS to process these records.

26.12 DS

The DS DNSSEC record type is fully supported, as described in [RFC 4034](#). Enabling DNSSEC for domains can be done with *pdnsutil*.

26.13 HINFO

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

26.14 KEY

The KEY record is fully supported. For its syntax, see [RFC 2535](#).

26.15 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

26.16 MX

The MX record specifies a mail exchanger host for a domain. Each mail exchanger also has a priority or preference. For example 10 mx.example.net. In the generic SQL backends, the 10 should go in the 'priority field'.

26.17 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.

26.18 NS

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

26.19 NSEC, NSEC3, NSEC3PARAM

The NSEC, NSEC3 and NSEC3PARAM DNSSEC record type are fully supported, as described in [RFC 4034](#). Enabling DNSSEC for domains can be done with *pdnsutil*.

26.20 OPENPGPKEY

The OPENPGPKEY records, specified in [RFC 7929](#), are used to bind OpenPGP certificates to email addresses.

26.21 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.

26.22 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`.

26.23 RRSIG

The RRSIG DNSSEC record type is fully supported, as described in [RFC 4034](#).

26.24 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. PowerDNS 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)

The fields have complicated and sometimes controversial meanings. The 'serial' field is special. If left at 0, the default, PowerDNS 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 Autoserial.

26.25 SPF

SPF records can be used to store Sender Policy Framework details ([RFC 4408](#)).

26.26 SSHFP

The SSHFP record type, used for storing Secure Shell (SSH) fingerprints, is fully supported. A sample from [RFC 4255](#) is:

```
2 1 123456789abcdef67890123456789abcdef67890
```

26.27 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.

26.28 TKEY, TSIG

The TKEY ([RFC 2930](#)) and TSIG records ([RFC 2845](#)), used for key-exchange and authenticated AXFRs, are supported. See the [TSIG](#) and *DNS update* <../dnsupdate> documentation for more information.

26.29 TLSA

Since 3.0. The TLSA records, specified in [RFC 6698](#), are used to bind SSL/TLS certificate to named hosts and ports.

26.30 SMIMEA

Since 4.1. The SMIMEA record type, specified in [RFC 8162](#), is used to bind S/MIME certificates to domains.

26.31 TXT

The TXT field can be used to attach textual data to a domain. Text is stored plainly, PowerDNS understands content not enclosed in quotes. However, all quotes characters (") in the TXT content must be preceded with a backslash (\):

```
"This \"is\" valid"
```

For a literal backslash in the TXT record, escape it:

```
"This is also \\ valid"
```

Unicode characters can be added in two ways, either by adding the character itself or the escaped variant to the content field. e.g. "ç" is equal to "\195\167".

When a TXT record is longer than 255 characters/bytes (excluding possible enclosing quotes), PowerDNS will cut up the content into 255 character/byte chunks for transmission to the client.

26.32 URI

The URI record, specified in [RFC 7553](#), is used to publish mappings from hostnames to URIs.

26.33 Other types

The following, rarely used or obsolete record types, are also supported:

- A6 ([RFC 2874](#), obsolete)
- DHCID ([RFC 4701](#))
- DLV ([RFC 4431](#))
- EUI48/EUI64 ([RFC 7043](#))
- IPSECKEY ([RFC 4025](#))
- KEY ([RFC 2535](#), obsolete)
- KX ([RFC 2230](#))
- MAILA ([RFC 1035](#))
- MAILB ([RFC 1035](#))
- MINFO ([RFC 1035](#))
- MR ([RFC 1035](#))
- RKEY (draft-reid-dnsext-rkey-00.txt)
- SIG ([RFC 2535](#), obsolete)
- WKS ([RFC 1035](#))

HTTP ROUTING TABLE

/servers	192
GET /servers, 185	PATCH /servers/{server_id}/zones/{zone_id}, 186
GET /servers/{server_id}, 185	
GET /servers/{server_id}/search-data, 192	
GET /servers/{server_id}/search-log, 193	
GET /servers/{server_id}/statistics, 193	
GET /servers/{server_id}/zones, 185	
GET /servers/{server_id}/zones/{zone_id}, 186	
GET /servers/{server_id}/zones/{zone_id}/check, 187	
GET /servers/{server_id}/zones/{zone_id}/cryptokeys, 189	
GET /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}, 190	
GET /servers/{server_id}/zones/{zone_id}/export, 187	
GET /servers/{server_id}/zones/{zone_id}/metadata, 191	
GET /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}, 191	
POST /servers/{server_id}/zones, 185	
POST /servers/{server_id}/zones/{zone_id}/cryptokeys, 190	
POST /servers/{server_id}/zones/{zone_id}/metadata, 191	
PUT /servers/{server_id}/zones/{zone_id}, 186	
PUT /servers/{server_id}/zones/{zone_id}/axfr-retrieve, 186	
PUT /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}, 190	
PUT /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}, 192	
PUT /servers/{server_id}/zones/{zone_id}/notify, 187	
PUT /servers/{server_id}/zones/{zone_id}/rectify, 187	
DELETE /servers/{server_id}/zones/{zone_id}, 186	
DELETE /servers/{server_id}/zones/{zone_id}/cryptokeys/{cryptokey_id}, 190	
DELETE /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind},	

D

DNSBackend::abortTransaction (C++ function), 356, 357
 DNSBackend::commitTransaction (C++ function), 356, 357
 DNSBackend::declare (C++ function), 352
 DNSBackend::declareArguments (C++ function), 352
 DNSBackend::feedRecord (C++ function), 356, 357
 DNSBackend::get (C++ function), 352
 DNSBackend::getArg (C++ function), 353
 DNSBackend::getArgAsNum (C++ function), 353
 DNSBackend::getSOA (C++ function), 352
 DNSBackend::getUpdatedMasters (C++ function), 356
 DNSBackend::list (C++ function), 351
 DNSBackend::listSubZone (C++ function), 356, 357
 DNSBackend::lookup (C++ function), 351
 DNSBackend::mustDo (C++ function), 353
 DNSBackend::replaceRRSet (C++ function), 357
 DNSBackend::setArgPrefix (C++ function), 353
 DNSBackend::setNotified (C++ function), 356
 DNSBackend::startTransaction (C++ function), 356, 357
 DNSResourceRecord (C++ class), 350
 DNSResourceRecord::auth (C++ member), 350
 DNSResourceRecord::content (C++ member), 350
 DNSResourceRecord::disabled (C++ member), 350
 DNSResourceRecord::domain_id (C++ member), 350
 DNSResourceRecord::last_modified (C++ member), 350
 DNSResourceRecord::qname (C++ member), 350
 DNSResourceRecord::qtype (C++ member), 350
 DNSResourceRecord::ttl (C++ member), 350
 DomainInfo (C++ class), 354
 DomainInfo::abortTransaction (C++ function), 354
 DomainInfo::backend (C++ member), 354
 DomainInfo::commitTransaction (C++ function), 354
 DomainInfo::feedRecord (C++ function), 354
 DomainInfo::getDomainInfo (C++ function), 354
 DomainInfo::getUnfreshSlaveInfos (C++ function), 354
 DomainInfo::id (C++ member), 354
 DomainInfo::isMaster (C++ function), 354
 DomainInfo::kind (C++ member), 354
 DomainInfo::last_check (C++ member), 354
 DomainInfo::master (C++ member), 354
 DomainInfo::notified_serial (C++ member), 354

DomainInfo::serial (C++ member), 354
 DomainInfo::setFresh (C++ function), 354
 DomainInfo::startTransaction (C++ function), 354
 DomainKind (C++ enum), 354

J

JSON Objects
 Comment, 189
 Cryptokey, 191
 Record, 189
 RRSet, 188
 SearchResult, 193
 Server, 185
 Zone, 187

R

RFC
 RFC 1034#section-6.2.1, 344
 RFC 1035, 370
 RFC 1183, 368
 RFC 1876, 367
 RFC 1996, 9
 RFC 2136, 55, 57
 RFC 2136#section-3.2, 58
 RFC 2136#section-3.6, 55
 RFC 2181, 107
 RFC 2230, 370
 RFC 2308, 255
 RFC 2535, 367, 370
 RFC 2538, 366
 RFC 2845, 61, 369
 RFC 2874, 370
 RFC 2915, 367
 RFC 2916, 367
 RFC 2930, 369
 RFC 3928, 119
 RFC 4025, 370
 RFC 4033, 36
 RFC 4034, 36, 366–368
 RFC 4035, 36
 RFC 4255, 369
 RFC 4408, 368
 RFC 4431, 370
 RFC 4509, 36
 RFC 4592, 181
 RFC 4701, 370

- [RFC 5155](#), [36](#)
- [RFC 5155#section-10.3](#), [42](#)
- [RFC 5155#section-4](#), [42](#)
- [RFC 5155#section-6](#), [42](#)
- [RFC 5702](#), [36](#)
- [RFC 5933](#), [36](#)
- [RFC 6605](#), [36](#)
- [RFC 6672](#), [366](#)
- [RFC 6698](#), [369](#)
- [RFC 6781](#), [44](#)
- [RFC 6781#section-4](#), [75](#)
- [RFC 6781#section-4.1.1.1](#), [76](#)
- [RFC 6844](#), [366](#)
- [RFC 6891#section-6.2.3](#), [252](#)
- [RFC 6979](#), [37](#)
- [RFC 7043](#), [370](#)
- [RFC 7159](#), [184](#)
- [RFC 7344](#), [51](#), [75](#), [261](#)
- [RFC 7344#appendix-B](#), [75](#)
- [RFC 7344#section-3.1](#), [366](#)
- [RFC 7344#section-3.2](#), [366](#)
- [RFC 7553](#), [369](#)
- [RFC 7766#section-10](#), [252](#)
- [RFC 7929](#), [367](#)
- [RFC 8080](#), [36](#)
- [RFC 8162](#), [369](#)

S

- [SOAData \(C++ class\)](#), [351](#)
- [SOAData::db \(C++ member\)](#), [351](#)
- [SOAData::default_ttl \(C++ member\)](#), [351](#)
- [SOAData::domain_id \(C++ member\)](#), [351](#)
- [SOAData::expire \(C++ member\)](#), [351](#)
- [SOAData::hostmaster \(C++ member\)](#), [351](#)
- [SOAData::nameserver \(C++ member\)](#), [351](#)
- [SOAData::refresh \(C++ member\)](#), [351](#)
- [SOAData::retry \(C++ member\)](#), [351](#)
- [SOAData::serial \(C++ member\)](#), [351](#)