PowerDNS Authoritative Server Documentation

PowerDNS.COM BV

Jan 17, 2025

CONTENTS

CHAPTER

ONE

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 metadata). The backends range from database backends (*MySQL*, *PostgreSQL*) and *BIND zone files* 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 Open-Xchange/PowerDNS company can provide help or support you in private as well. Please contact PowerDNS.

1.2.1 My information is confidential, must I send it to the mailing list, discuss it on IRC, or post it in a GitHub ticket?

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.

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.

CHAPTER

TWO

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. Your distribution likely ships a package, but we recommend getting more recent packages from the PowerDNS repositories. Please see the instructions on the repo site and then come back here!

```
$ sudo apt-get install pdns-server
```

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

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

2.1.2 Redhat-based Systems

On RedHat based systems there are 2 options to install PowerDNS, from EPEL, or from the PowerDNS repositories:

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

```
$ sudo yum install pdns
```

The different backends can be installed using

```
$ sudo yum install pdns-backend-$backend
```

Note that for some of those package sources, the bind backend is shipped as part of the base `pdns` package, and there is no separate `pdns-backend-bind` package.

2.1.3 FreeBSD

PowerDNS Authoritative Server is available through the ports system:

For the package:

```
$ sudo 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, try *Basic setup: configuring database connectivity* using SQLite 3 or start *migrating* your data.

CHAPTER

THREE

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 4.9.0 to 5.0.0/master

3.1.1 LUA records whitespace insertion

lua-records-insert-whitespace, introduced in 4.9.1 with the default value (`yes`) set to maintain the old behaviour of inserting whitespace, is set to `no` in 5.0.

3.1.2 ixfrdist IPv6 support

`ixfrdist` now binds listening sockets with *IPV6_V6ONLY set*, which means that `[::]` no longer accepts IPv4 connections. If you want to listen on both IPv4 and IPv6, you need to add a line with `0.0.0.0` to the `listen` section of your ixfrdist configuration.

3.2 4.8.0 to 4.9.0

3.2.1 Removed options

Various settings, deprecated since 4.5.0, have been removed.

- *allow-unsigned-supermaster* is now *allow-unsigned-autoprimary*
- *master* is now *primary*
- *slave-cycle-interval* is now *xfr-cycle-interval*
- *slave-renotify* is now *secondary-do-renotify*
- *slave* is now *secondary*
- *superslave* is now *autosecondary*

In *lmdb-sync-mode*, the previous default `mapasync` is no longer a valid value. Due to a bug, it was interpreted as `sync` in previous versions. To avoid operational surprises, `sync` is the new default value.

3.2.2 Renamed options

Bind backend

Various experimental autoprimary settings have been renamed.

- `supermaster-config` is now `autoprimary-config`
- `supermasters` is now `autoprimaries`
- `supermaster-destdir` is now `autoprimary-destdir`

Gsql backends

Various custom queries have been renamed.

- `info-all-slaves-query` is now `info-all-secondaries-query`
- `supermaster-query` is now `autoprimary-query`
- `supermaster-name-to-ips` is now `autoprimary-name-to-ips`
- `supermaster-add` is now `autoprimary-add`
- `update-master-query` is now `update-primary-query`
- `info-all-master-query` is now `info-all-primary-query`

Also, `get-all-domains-query` got an extra column for a zone's catalog assignment.

API changes

A long time ago (in version 3.4.2), the `priority` field was removed from record content in the HTTP API. Starting with 4.9, API calls containing a `priority` field are actively rejected. This makes it easier for users to detect they are attempting to use a very old API client.

3.3 any version to 4.8.x

3.3.1 Use of (RSA-)SHA1 on Red Hat Enterprise Linux 9 and derivatives

If you are using PowerDNS Authoritative Server on EL9, please read this ticket about Red Hat's SHA1 deprecation and how it affects PowerDNS software.

3.3.2 LMDB backend

Version 4.8.0-alpha1 ships a new version of the LMDB database schema (called version 5), for compatibility with Lightning Stream. This schema is somewhat experimental, and although we do intend to make databases portable/upgradeable to future releases in the 4.8 train, we currently make no promises. There is no downgrade process. If you upgrade your database (by starting 4.8.0 without `lmdb-schema-version=4`), you cannot go back.

Upgrading is only supported from database schema versions 3 and 4, that is, databases created/upgraded by version 4.4 and up.

In version 4.8.0, schema version 5 is finalised. Databases created with -alpha1 or -beta1 work with 4.8.0.

3.4 4.6.0 to 4.7.0

3.4.1 Schema changes

The new Catalog Zones feature comes with a mandatory schema change for the gsql database backends. See files named `4.3.x_to_4.7.0_schema.X.sql` for your database backend in our Git repo, tarball, or distro-specific documentation path. For the LMDB backend, please review *lmdb-schema-version*. The new LMDB schema version is 4.

3.5 4.5.x to 4.6.0

3.5.1 Automatic conversion of `@` signs in SOA

Before version 4.5.0, PowerDNS would automatically replace `@` in the SOA RNAME with `.`, making it easy for users to enter their hostmaster email address without having to think about syntax. However, this feature interacts badly with handling of presigned zones. In version 4.5.0, this feature was accidentally broken in the implementation of the zone cache. In 4.6.0, this automatic conversion is fully removed. If you still have `@` signs in any SOA RNAMEs, 4.6.0 will serve those out literally. You can find any stray `@` signs by running `pdnsutil check-all-zones`.

3.5.2 New default NSEC3 parameters

Following draft-ietf-dnsop-nsec3-guidance (Guidance for NSEC3 parameter settings), the default NSEC3PARAM settings (see *Setting the NSEC modes and parameters*) in pdnsutil are now *1 0 0 -* instead of *1 0 1 ab*.

3.5.3 SHA1 DSes

`pdnsutil show-zone` and `pdnsutil export-zone-ds` no longer emit SHA1 DS records, unless `--verbose` is in use.

3.5.4 Privileged port binding in Docker

In our Docker image, our binaries are no longer granted the `net_bind_service` capability, as this is unnecessary in many deployments. For more information, see the section "Privileged ports" in Docker-README.

3.6 4.4.x to 4.5.0

3.6.1 Automatic conversion of `@` signs in SOA

Before version 4.5.0, PowerDNS would automatically replace `@` in the SOA RNAME with `.`, making it easy for users to enter their hostmaster email address without having to think about syntax. In version 4.5.0, this feature was accidentally broken in the implementation of the zone cache, and the replacement would only happen if the zone cache was disabled. Note that in 4.6.0, this automatic conversion is fully removed. If you still have `@` signs in any SOA RNAMEs, 4.5.0 will serve those out literally if the zone cache is enabled.

3.6.2 Record type changes

The in-database format of `CSYNC`, `IPSECKEY`, `NID`, `L32`, `L64`, and `LP` records has changed from 'generic' format to its specialized format.

Generation of the in-database format of `SVCB` and `HTTPS` received some important bug fixes. (For these two types, you can skip the *upgrade-unknown-types* setting mentioned below, but we still recommend the re-transfer.)

API users might notice that replacing records of the newly supported types leaves the old TYPExx records around, even if PowerDNS is not serving them. To fix this, enable *upgrade-unknown-types* and replace the records; this will then delete those TYPExx records. Then, disable the setting again, because it has a serious performance impact on API operations.

On secondaries, it is recommended to re-transfer, using `pdns_control retrieve ZONE`, with *upgrade-unknown-types* enabled, all zones that have records of those types, or `TYPExx`, for numbers 45 and 62. Leave the setting on until all zones have been re-transferred.

3.6.3 Changed options

Renamed options

Various settings have been renamed. Their old names still work in 4.5.x, but will be removed in a release after it.

- *allow-unsigned-supermaster* is now *allow-unsigned-autoprimary*
- *master* is now *primary*
- *slave-cycle-interval* is now *xfr-cycle-interval*
- *slave-renotify* is now *secondary-do-renotify*
- *slave* is now *secondary*
- *superslave* is now *autosecondary*
- *domain-metadata-cache-ttl* is now *zone-metadata-cache-ttl*

Changed defaults

- The default value of the *consistent-backends* option has been changed from `no` to `yes`.
- The default value of the *max-nsec3-iterations* option has been changed from `500` to `100`.
- The default value of the `timeout` parameter for *`ifportup()`* and *`ifurlup()`* functions has been changed from `1` to `2` seconds.
- The default value of the new *zone-cache-refresh-interval* option is `300`.

Zone cache

Version 4.5 introduces the zone cache. The default refresh interval (*zone-cache-refresh-interval*) is 300, meaning that zones newly added to your backend may need a few minutes to appear. However, zones added using the API should not notice a delay.

If your backend is dynamic in what zones it does or does not offer, and thus cannot easily provide a complete list of zones every few minutes, set the interval to 0 to disable the feature.

Removed options

- *local-ipv6* has been removed. IPv4 and IPv6 listen addresses should now be set with *local-address*.
- *query-local-address6* has been removed. IPv4 and IPv6 addresses used for sending queries should now be set with *query-local-address*.

3.7 4.3.x to 4.4.0

3.7.1 Latency calculation changes

It turned out that average latency calculations in earlier versions used integers instead of floating point variables, which led to the throwing away of any data points between 'the current average' and 1000ms above it, instead of having those data points affecting the average. In 4.3.2 and 4.4.0, we started using floating point variables for this, which means the latency calculation is accurate now. Usually, this means you will see higher latency numbers after upgrading.

3.7.2 MySQL character set detection

Before 4.4.0, the gmysql backend told the MySQL (or MariaDB) client libraries to automatically detect the client character set and collation, based on the environment locale. (Look for 'autodetect' in https://dev.mysql.com/doc/refman/5.7/en/charset-connection.html to know more). On some systems, this autodetection makes choices that are incompatible with MySQL Server 8 defaults. On all systems, this autodetection can make choices that vary depending on how PowerDNS is started. In other words, the autodetection provides unpredictable results.

In 4.4.0, the autodetection has been removed. The MySQL/MariaDB client lib will now use its default settings, unless overridden in `my.cnf`, for example:

```
[client]
default-character-set = latin1
```

If you have trouble connecting to your database with 4.4.0 or up, you can override the character set in `my.cnf`.

Before upgrading, please check your database for any non-ASCII content. The interpretation of the non-ASCII bytes in those fields might change because of a different charset suddenly being used.

3.7.3 Record type changes

The in-database format of the `SVCB`, `HTTPS` and `APL` records has changed from 'generic' format to its specialized format.

API users might notice that replacing records of these types leaves the old TYPExx records around, even if PowerDNS is not serving them. To fix this, enable *upgrade-unknown-types* and replace the records; this will then delete those TYPExx records. Then, disable the setting again, because it has a serious performance impact on API operations.

On secondaries, it is recommended to re-transfer, using `pdns_control retrieve ZONE`, with *upgrade-unknown-types* enabled, all zones that have records of those types, or `TYPExx`, for numbers 42, 64, 65. Leave the setting on until all zones have been re-transferred.

3.7.4 PostgreSQL configuration escaping

We now correctly quote/escape Postgres connection parameters. If you used single quotes (or some other form of escaping) around your Postgres password because it contained spaces, you now need to put your unmodified, unescaped, unquoted password in your configuration.

3.7.5 New LMDB schema

An LMDB schema upgrade is mandatory. Please carefully read *lmdb-schema-version* before upgrading to 4.4.x. The new schema version is version 3.

3.7.6 Removed features

SOA autofilling (i.e. allowing incomplete SOAs in the database) and the API `set-ptr` feature, that both were deprecated in earlier releases, have now been removed. Please update your configuration and remove the following settings:

- *default-soa-mail*
- *default-soa-name*
- *soa-expire-default*
- *soa-minimum-ttl*
- *soa-refresh-default*
- *soa-retry-default*

Replace them with *default-soa-content*, but be aware that this will only be used at zone creation time. Please run `pdnsutil check-all-zones` to check for incomplete SOAs.

The *do-ipv6-additional-processing* setting was removed. IPv6 additional processing now always happens when IPv4 additional processing happens.

3.8 4.3.1 to 4.3.2

3.8.1 Latency calculation changes

It turned out that average latency calculations in earlier versions used integers instead of floating point variables, which led to the throwing away of any data points between ‘the current average’ and 1000ms above it, instead of having those data points affecting the average. In 4.3.2 and 4.4.0, we started using floating point variables for this, which means the latency calculation is accurate now. Usually, this means you will see higher latency numbers after upgrading.

To be very clear, there is no performance difference between 4.3.1 and 4.3.2. The only change is in the latency calculation, which was wrong in 4.3.1 and is correct in 4.3.2. This fix was backported to 4.3.2 from 4.4.0 so that users can fairly compare the performance of 4.3.2 and 4.4.0.

3.9 4.3.0 to 4.3.1

On RHEL/CentOS 8, the gmysql backend now uses `mariadb-connector-c` instead of `mysql-libs`. This change was made because the default MySQL implementation for RHEL8 is MariaDB, and MariaDB and MySQL cannot be installed in parallel due to conflicting RPM packages. The mariadb client lib will connect to your existing MySQL servers without trouble.

Unknown record encoding (RFC 3597) has become more strict as a result of the fixes for *PowerDNS Security Advisory 2020-05*. Please use `pdnsutil check-all-zones` to review your zone contents.

The previous set of indexes for the gsqlite3 backend was found to be poor. 4.3.1 ships a new schema, and a migration:

```
.bail on

CREATE INDEX records_lookup_idx ON records(name, type);
CREATE INDEX records_lookup_id_idx ON records(domain_id, name, type);
CREATE INDEX records_order_idx ON records(domain_id, ordername);

DROP INDEX IF EXISTS rec_name_index;
DROP INDEX IF EXISTS nametype_index;
DROP INDEX IF EXISTS domain_id;
```

(continues on next page)

(continued from previous page)

```
DROP INDEX IF EXISTS orderindex;

CREATE INDEX comments_idx ON comments(domain_id, name, type);

DROP INDEX IF EXISTS comments_domain_id_index;
DROP INDEX IF EXISTS comments_nametype_index;

ANALYZE;
```

3.10 4.2.x to 4.3.0

3.10.1 NSEC(3) TTL changed

NSEC(3) records now use the negative TTL, instead of the SOA minimum TTL. See *the DNSSEC TTL notes* for more information.

3.10.2 Lua Netmask class methods changed

Netmask class methods `isIpv4` and `isIpv6` have been deprecated in Lua, use `Netmask.isIPv4()` and `Netmask.isIPv6()` instead. In the C++ API, these methods have been removed.

3.10.3 `socket-dir` changed

The default *socket-dir* has changed to include `pdns` in the path. It is now whatever is passed to `--with-socketdir` during configure (`/var/run` by default) plus `pdns`. The systemd unit-file is updated to reflect this change and systemd will automatically create the directory with the proper permissions. The packaged sysV init-script also creates this directory. For other operating systems, update your init-scripts accordingly.

3.10.4 Systemd service and permissions

The systemd service-file that is installed no longer uses the `root` user to start. It uses the user and group set with the `--with-service-user` and `--with-service-group` switches during configuration, “pdns” by default. This could mean that PowerDNS cannot read its configuration or zone-file data. It is recommended to recursively `chown` directories used by PowerDNS:

```
# For Debian-based systems
chown -R root:pdns /etc/powerdns
chown -R pdns:pdns /var/lib/powerdns

# For CentOS and RHEL based systems
chown -R root:pdns /etc/pdns
chown -R pdns:pdns /var/lib/pdns
```

Packages provided on the PowerDNS Repository will `chown` directories created by them accordingly in the post-installation steps.

3.10.5 New settings

- The *axfr-fetch-timeout* setting has been added. This setting controls how long an inbound AXFR may be idle in seconds. Its default is 10
- The *max-generate-steps* setting has been added. This sets the maximum number of steps that will be performed when loading a BIND zone with the `$GENERATE` directive. The default is 0, which is unlimited.

3.10.6 Deprecated settings

- *local-ipv6* has been deprecated and will be removed in 4.5.0. Both IPv4 and IPv6 listen addresses can now be set with *local-address*. The default for the latter has been changed to `0.0.0.0, ::`.

3.10.7 Changed defaults

- *local-address* now defaults to `0.0.0.0, ::`.

3.10.8 Schema changes

- The new 'unpublished DNSSEC keys' feature comes with a mandatory schema change for all database backends (including BIND with a DNSSEC database). See files named `4.2.0_to_4.3.0_schema.X.sql` for your database backend in our Git repo, tarball, or distro-specific documentation path. For the LMDB backend, please review *lmdb-schema-version*.
- If you are upgrading from 4.3.0-beta2 or 4.3.0-rc2, AND ONLY THEN, please read pull request #8975 very carefully.

3.10.9 Implicit 5->7 algorithm upgrades

Since version 3.0 (the first version of the PowerDNS Authoritative Server that supported DNSSEC signing), we have automatically, silently, upgraded algorithm 5 (RSASHA1) keys to algorithm 7 (RSASHA1-NSEC3-SHA1) when the user enabled NSEC3. This has been a source of confusion, and because of that, we introduced warnings for users of this feature in 4.0 and 4.1. To see if you are affected, run `pdnsutil check-all-zones` from version 4.0 or up. In this release, the automatic upgrade is gone, and affected zones will break if no action is taken.

3.10.10 IXFR-in corruption

A bug in PowerDNS versions before 4.2.2/4.3.0 would cause wrong deletion or addition of records if IXFR deltas came in very quickly (within the query cache timeout, which defaults to 20/60 seconds). If you have zones which use inbound IXFR (in other words, the `IXFR` metadata item for that zone is set to `1`), we strongly suggest triggering a completely fresh transfer. You could accomplish that by deleting all records in the zone with an SQL query and waiting for a fresh transfer, or (1) disabling IXFR (2) forcing a fresh transfer using `pdns_control retrieve example.com` (3) enabling IXFR again.

3.11 4.2.X to 4.2.3

Unknown record encoding (RFC 3597) has become more strict as a result of the fixes for *PowerDNS Security Advisory 2020-05*. Please use `pdnsutil check-all-zones` to review your zone contents.

3.12 4.X.X to 4.2.2

3.12.1 IXFR-in corruption

A bug in PowerDNS versions before 4.2.2/4.3.0 would cause wrong deletion or addition of records if IXFR deltas came in very quickly (within the query cache timeout, which defaults to 20/60 seconds). If you have zones which use inbound IXFR (in other words, the `IXFR` metadata item for that zone is set to `1`), we strongly suggest triggering a completely fresh transfer. You could accomplish that by deleting all records in the zone with an SQL query and waiting for a fresh transfer, or (1) disabling IXFR (2) forcing a fresh transfer using `pdns_control retrieve example.com` (3) enabling IXFR again.

3.13 4.1.X to 4.2.0

- Superslave operation is no longer enabled by default, use *superslave* to enable. This setting was called `supermaster` in some 4.2.0 prereleases.
- The gsqlite3 backend, and the DNSSEC database for the BIND backend, have a new journal-mode setting. This setting defaults to WAL; older versions of PowerDNS did not set the journal mode, which means they used the SQLite default of DELETE.
- Autoserial support has been removed. The `change_date` column has been removed from the `records` table in all gsql backends, but leaving it in is harmless.
- The *Generic PostgreSQL backend* schema has changed: the `notified_serial` column type in the `domains` table has been changed from `INT DEFAULT NULL` to `BIGINT DEFAULT NULL`: `ALTER TABLE domains ALTER notified_serial TYPE bigint USING CASE WHEN notified_serial >= 0 THEN notified_serial::bigint END;`
- Rectification after API changes is now default (*default-api-rectify*). If you do mutations in large zones, you may notice a slowdown.

3.14 4.1.X to 4.1.14

Unknown record encoding (RFC 3597) has become more strict as a result of the fixes for *PowerDNS Security Advisory 2020-05*. Please use `pdnsutil check-all-zones` to review your zone contents.

3.15 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;`

3.16 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

3.16.1 Changed options

- `experimental-lua-policy-script` option and the feature itself have been completely dropped. We invite you to use PowerDNS dnsdist 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`.

3.16.2 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 `--with-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.

3.17 4.0.X to 4.0.2

3.17.1 Changed options

Changed defaults

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

3.18 3.4.X to 4.0.0

3.18.1 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 ordername (or NULL)
- `insert-slave-query` has been dropped, `insert-zone-query` now sets the type of zone

3.18.2 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`

3.18.3 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).

3.18.4 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.

CHAPTER

FOUR

DNS MODES OF OPERATION

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

4.1 Native replication

Native replication is the default unless another operation is specifically configured. Native replication means that PowerDNS will not send out DNS update notifications, nor will it 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.

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

Typically, a database secondary 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 primary or secondary in zone transfers, or has a frontend attached for managing records, etc.

4.2 Primary operation

When operating as a primary, PowerDNS sends out notifications of changes to secondaries, 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 signal zone changes to secondaries, as described in **RFC 1996**. Since version 4.0.0, the NOTIFY messages have a TSIG record added (transaction signature) if the zone has been configured to use TSIG and the feature has been enabled.

Warning: Primary support is OFF by default, turn it on by adding *primary* to the configuration. You also need to set the type of the zones to be served as primary, see next warning.

Warning: Notifications are only sent for domains with type PRIMARY or MASTER in your backend unless *secondary-do-renotify* is enabled.

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

Left open by **RFC 1996** is who is to be notified - which is harder to figure out than it sounds. All secondaries for this domain must receive a notification but the nameserver only knows the names of the secondaries - 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 secondaries 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 choose 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 secondaries 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 secondaries 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 Secondary operation

On launch, PowerDNS requests from all backends a list of domains that 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 primary server. If the *SOA* serial number there is higher, the domain is retrieved and updated in the database. In any case, after the check, the domain is declared 'fresh', and will only be checked again after '**refresh**' seconds have passed.

If the serial is equal, PowerDNS as a secondary with a presigned zone will also compare the SOA RRSIG (signature). If the signatures are different, the zone is also queued for a zone transfer. This is useful when the primary server updates DNSSEC signatures without changing the zone serial. In some configurations, a PowerDNS primary can exhibit this behaviour. To allow for this check, the DO flag is set on the SOA query towards the primary server. In some conditions, some primary servers answer with a truncated SOA response (indicating TCP is required), and the freshness check will fail. As a workaround, the signature check and DO flag can be turned off by disabling *secondary-check-signature-freshness*.

When the freshness of a domain cannot be checked, e.g. because the primary is offline, PowerDNS will retry the domain after *xfr-cycle-interval* seconds. Every time the domain fails its freshness check, PowerDNS will hold back on checking the domain for `amount of failures * xfr-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. The same hold back algorithm is also applied if the zone transfer fails due to problems on the primary, i.e. if zone transfer is not allowed.

Receiving a NOTIFY immediately clears the back-off period for the respective domain to allow immediate freshness checks for this domain.

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

Warning: Only domains with type SECONDARY or SLAVE are considered for secondary support.

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'd 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 does PostgreSQL. The BIND backend implements transaction semantics by renaming files if and only if they have been retrieved completely and parsed correctly.

Secondary 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 primary.

Since 4.5.0, zone transfers are added to a queue and processed according to priority and order of addition. Order levels are (from high to low): pdns control, api, notify, serial changed during refresh and signatures changed during refresh. High priority zone transfers are always processed first, in a first in first out order.

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

Since version 4.0.0, PowerDNS requires that primaries 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 Primary/Secondary Setup Requirements

Generally to enable a Primary/Secondary setup you have to take care of the following properties.

- The *primary*/*secondary* 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 secondary.
- Primary/Secondary 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 secondary needs a comma-separated list of primary 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 secondary zone changes from non-DNSSEC to DNSSEC, an IXFR update will not set the PRESIGNED 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 cannot serve IXFR updates.

4.6 Autoprimary: automatic provisioning of secondaries

Changed in version 4.5.0: Before version 4.5.0, this feature was called 'supermaster'

PowerDNS can recognize so-called 'autoprimaries'. An autoprimary is a host which is primary for domains and for which we are to be a secondary. When a primary (re)loads a domain, it sends out a notification to its secondaries. Normally, such a notification is only accepted if PowerDNS already knows that it is a secondary for a domain.

However, a notification from an autoprimary carries more persuasion. When PowerDNS determines that a notification comes from an autoprimary and it is bonafide, it can provision the domain automatically, and configure itself as a secondary for that zone.

Before an autoprimary notification succeeds, the following conditions must be met:

- *autosecondary* support must be enabled
- The autoprimary must carry a SOA record for the notified domain
- The autoprimary IP must be present in the `supermasters` table in the database on the secondary, along with any name that is in the NS set.
- The set of NS records for the domain, as retrieved by the secondary from the autoprimary, must include the name that goes with the IP address in the `supermasters` table
- If your primary sends signed NOTIFY it will mark that TSIG key as the TSIG key used for retrieval as well
- If you turn off *allow-unsigned-autoprimary*, then your autoprimaries are required to sign their notifications.

> **Warning:** If you use another PowerDNS server as primary 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 autoprimary, and how PowerDNS will be listed in the set of NS records remotely, and the 'account' name of your autoprimary. There is no need to fill the account name out but it does help keep track of where a domain comes from. Additionally, if a secondary selects multiple autoprimaries for a zone based on the name of the primary, it also checks that the `account` field is the same for all. Adding a autoprimary can be done either directly in the database, or by using the 'pdnsutil add-autoprimary' command.

> **Warning:** When a secondary receives notification while bootstrapping a new domain using autosecondary feature, it will send SOA and NS queries to the IP address matched in the `supermasters` table. These queries are **not** recursive. This will cause domain bootstrap to fail if the primary authoritative server is hidden behind a recursor, so make sure these queries go (or are forwarded by dnsdist) straight to the auth server.

> **Note:** Removal of zones provisioned using the autoprimary must be done on the secondaries themselves, as there is no way to signal this removal from the primary to the secondary.

4.7 Modifying a secondary 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

This script can be enabled like this:

```
pdnsutil set-meta example.com LUA-AXFR-SCRIPT /path/to/lua/script.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 a 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 record 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 timestamp
   if record:qtype() == pdns.TXT and string.starts(record:qname():toString(), "_
↪tstamp.") then
      resp = {}
      resp[1] = {
        qname   = record:qname():toString(),
        qtype   = record:qtype(),
        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
```

(continues on next page)

(continued from previous page)

```
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 timestamp. A records are appended with a TXT record. All other records are unhandled.

CHAPTER

FIVE

MIGRATING TO POWERDNS

Before migrating to PowerDNS a few things should be considered.

PowerDNS does not operate as a *secondary* or *primary* 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 *secondary* 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 *primary* 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 *primary* 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 output, 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* 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.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. This include the SOA record too.

5.3 Migrating Data from one Backend to Another Backend

Note: This is an 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.

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.

CHAPTER

SIX

RUNNING AND OPERATING

On Linux, PowerDNS is controlled by a systemd service called `pdns.service`. The service definition file should be installed by the binary package, and can also be found in the tarball (`pdns.service.in` template file).

On non-Linux systems, a SysV-style init script can be used, and should be supplied by the operating system packages.

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

Also see *Running Virtual Instances*.

6.1 Guardian

When the init-system of the Operating System does not properly supervises 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 *Primary operation*.

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

6.4.3 Backend manipulation

6.4.4 `pdnsutil`

To perform zone and record changes using inbuilt tools, the `pdnsutil` command can be used. All available options are described in the online *manual page* as well as in `man pdnsutil`.

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.

CHAPTER

SEVEN

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 peter.van.dijk@powerdns.com and remi.gacogne@powerdns.com. In case you want to encrypt your report using PGP, please use: https://doc.powerdns.com/powerdns-keyblock.asc

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 and dnsdist users that under the terms of the GNU General Public License, PowerDNS and dnsdist come with ABSOLUTELY NO WARRANTY. This *license* is included in this documentation.

If you believe you have found a security vulnerability that applies to DNS implementations generally, and you want to report this responsibly to a number of implementers, you might consider also using the Open Source DNS Vulnerability mailing list, managed by DNS-OARC.

7.1.1 YesWeHack

Security issues can also be reported on our YesWeHack page and might fetch a bounty. Do note that only the PowerDNS software is in scope for the YesWeHack 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

Modern Linux distributions, with systemd for process management, do a better job of constraining PowerDNS than chroot can. We strongly suggest using distribution/OS features for process containment instead of the *chroot* option. The text below is kept for those users that have specific reasons to prefer chroot. chroot functionality is not actively tested during development and might break during upgrades.

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 primary or secondary 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 empathically 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.

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 finds 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. Case 2 can also be issued for non-security related upgrade recommendations for pre-releases. 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 greater 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 cannot 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.

7.5 Trusting zone files

In some scenarios the PowerDNS server must handle zone files coming from an untrusted third party. For these cases, it is recommended to take extra protective measures in addition to the measures above:

- Set *max-generate-steps* to a low number, this will limit the amount of resources used by rogue `$GENERATE` templates.
- Set *max-include-depth* to `0`, this will disallow the `$INCLUDE` directive, avoiding problems with include loops and related issues.
- Set *enable-lua-records* to `no`, this will disable *Lua Records*.

Depending on your specific requirements, it might be good perform checks on zone files before loading the zone into PowerDNS to:

- Enforce reasonable `TTL` values.
- Enforce reasonable values in the `SOA` records.
- Validate delegations.
- Enforce a reasonable maximum for the total number of records.
- Enforce a reasonable maximum for the number of records per record set.

CHAPTER

EIGHT

PERFORMANCE AND TUNING

In general, best performance is achieved on recent Linux kernels with the bindbackend, or if something more database-like is preferred, the LMDB backend. Meanwhile many of the largest PowerDNS installations are based on PostgreSQL or MySQL.

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. This number is a total over all receiver threads.

The *max-queue-length* and *overload-queue-length* settings determine how PowerDNS deals with growing queues. If the queue for a single receiver thread (and its associated distributor threads) grows beyond the `overload` number, queries are answered only from the packet cache so the database can hopefully recover. If we reach the `max` number, we consider the situation hopeless and respawn the server process.

The value of *queue-limit* should be set to only keep queries in queue for as long as someone would be interested in knowing the answer. Many resolvers will query other name servers for the zone quite aggressively.

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.

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 zero internal processing is done when answering a question from the Packet Cache.

8.4 Caches & Memory Allocations & glibc

Managing the two caches described above involves a lot of memory management, that is handled by `malloc` in your libc. To avoid contention between threads, the allocator in glibc separates memory into separate arenas, sometimes even hundreds of them. This avoids locking, but it may cause massive memory fragmentation, that could make PowerDNS take an order of magnitude more memory in some situations.

If you suspect this is happening on your setup, you can consider lowering `MALLOC_ARENA_MAX` to a small number. Several users have reported that `4` works well for them. Via `systemctl edit pdns` you can put `Environment=MALLOC_ARENA_MAX=4` in your pdns unit file to enable this tweak.

Note that newer glibc versions replace MALLOC_ARENA_MAX with a different setting syntax. The new syntax is `GLIBC_TUNABLES=glibc.malloc.arena_max=4`, please check which syntax is valid for your glibc version (it is quite likely that both syntaxes will work).

8.5 Performance Monitoring

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

8.5.1 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

open-tcp-connections

Number of currently open TCP connections

overload-drops

Number of questions dropped because backends overloaded (backends are overloaded if they have more outstanding queries than the value of *overload-queue-length*)

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, only available if *distributor-threads* > 1

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).

receive-latency

Average number of microseconds needed to receive a query

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*.

send-latency

Average number of microseconds needed to send the answer

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-csum-errors

Number of UDP packets received with an invalid checksum

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-in-csum-errors

Number of IPv6 UDP packets received with an invalid checksum

udp6-in-errors

Number of IPv6 UDP packets received faster than the OS could process them

udp6-noport-errors

Number of IPv6 UDP packets where an ICMP response was received that the remote port was not listening

udp6-queries

Number of questions received over UDPv6

udp6-recvbuf-errors

Number of errors caused in the IPv6 UDP receive buffer

udp6-sndbuf-errors

Number of errors caused in the IPv6 UDP send buffer

uptime

Uptime in seconds of the daemon

user-msec

Number of milliseconds spend in CPU ‘user’ time

8.5.2 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.

8.5.3 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.

CHAPTER

NINE

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 PowerDNS installation with DNSSEC.

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 obtained 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)
- 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)
- 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 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. If intercepted, keying material 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.

Please note that the ALIAS record type is *not supported* in this mode.

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 deactivated. Subsequently, it can be removed.

As described above, there are several ways in which DNSSEC can deny the existence of a record, and this setting, which is also stored away from zone records, 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. NSEC is the default mode for secured zones in PowerDNS.

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. This is the default mode for NSEC3 in PowerDNS.

NSEC3 in ‘narrow’ mode uses additional hashing calculations to provide hashed secure denial-of-existence ‘on the fly’ per RFC 7129, without further involving the database. This mode will make PowerDNS to send out “white lies” and prevents zone enumeration, but these responses require online signing capabilities by all nameservers and therefore denies incoming AXFRs for zones in this mode.

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, the RRSIG validity 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:

```
RRSIG(1) Inception                                                    RRSIG(1)␣
↪Expiry
|                                                                                ␣
↪ |
v                                                                                ␣
↪ v
|================================ RRSIG(1) validity␣
↪=================================|
                            |================================ RRSIG(2) validity␣
↪=================================|
                            ^                                                    ␣
↪                             ^
                            |                                                    ␣
↪                             |
                            RRSIG(2) Inception                                   ␣
↪                RRSIG(2) Expiry

                            |----- RRSIG(1) served -----|----- RRSIG(2) served ----
↪-|

|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--
↪-|---|---|---|---|---|---|---|
Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue␣
↪Wed Thu Fri Sat Sun Mon Tue Wed Thu
                                                        ^
                                                        |
                                                        RRSIG roll-over(1 to 2)
```

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 the integral number of weeks 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 served as secondary from a remote primary in online signing mode, or can be pre-signed using tools like OpenDNSSEC, ldns-signzone, and 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 secondary to an unsigned primary.

In this way, if keying material is available for an unsigned zone that is retrieved from a primary 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 primary 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 primary.

The signing and hashing algorithms are described in *Online Signing*.

9.3.5 BIND-mode operation

The *BIND backend* can manage keys and other DNSSEC-related *domain metadata* in an SQLite3 database without launching a separate gsqlite3 backend.

To use this mode, run `pdnsutil create-bind-db /var/lib/powerdns/bind-dnssec-db.sqlite3` and set *bind-dnssec-db* in pdns.conf to the path of the created database. 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 ‘BIND backend’ 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 ‘BIND backend’ 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 CSK. 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] ['narrow']
```

e.g.

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

The quoted part is the content of the NSEC3PARAM records, as defined in **RFC 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. Limited by the *max-nsec3-iterations* setting.
- Salt to apply during hashing, in hexadecimal, or `-` to use no salt

Optionally, NSEC3 can be set to 'narrow' mode. For more information refer to *(Hashed) Denial of Existence*.

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 secondaries

As RRSIGs can expire, secondary 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 primary.

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 *primary* zones (where replication happens by means of AXFR), PowerDNS secondaries 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 secondaries, the secondaries 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.

EPOCH

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

Warning: Don't combine this with AXFR - the secondaries 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 secondaries)

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.

A slightly more complex approach is running a *hidden* primary in simple online signing mode, but on a highly secured system unreachable for the public. Internet-connected secondaries can then transfer the zones pre-signed from this primary over a secure private network. This topology offers substantial security benefits with regards to key material while maintaining ease of daily operation by PowerDNS's features in online mode.

See also *Pre-signed records*.

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**.

9.6.7 Some notes on TTL usage

In zones signed by PowerDNS (so non-presigned zones), some TTL values need to be filled in by PowerDNS. The TTL of RRSIG record sets is the TTL of the covered RRset. For CDS, CDNSKEY, DNSKEY, NSEC, NSEC3 and NSEC3PARAM, we use the SOA minimum (the last number in the SOA record). Except for CDS/CDNSKEY/DNSKEY, these TTLs are chosen because RFC 4034 demands it so.

If you want a 'normal' TTL (3600, 86400, etc.) for your DNSKEY but a low TTL on negative answers, set your SOA minimum TTL to the high number, and set the TTL on the SOA record itself to the low TTL you want for negative answers. Note that the NSEC/NSEC3 records proving those negatives will get the high TTL in that case, and this may affect subsequent resolution in resolvers that do aggressive NSEC caching (RFC 8198).

Note: This behaviour was changed in version 4.3.0. We believe the language in RFC 4034 and 5155 about the NSEC(3) TTL is a mistake, and we have chosen to honour its spirit instead of its words.

This unfortunate wording was eventually corrected in **RFC 9077**.

NSEC(3) records now get the negative TTL (which is the lowest of the SOA TTL and the SOA minimum), which means their TTL matches that of an error such as NXDOMAIN. This conforms to RFC9077.

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 easy to (mis)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 primary/secondary setup, you may need to tinker with the *SOA-EDIT* metadata on your primary. This is described in the *SOA-EDIT: ensure signature freshness on secondaries* 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 is an experimental feature, use at your own risk!

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

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

9.8.1 Using PKCS#11 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`. In 4.2.0, Botan support has been removed and this is no longer an issue.

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

These instructions have been tested on Debian 10 (Buster).

- `apt-get install softhsm p11-kit`
- Verify that it works: `p11-kit -l`, you should see `softhsm2: .....`
- Create a token:

```
softhsm2-util --init-token --label my-pkcs11-dnskey --free --pin 1234 --so-pin␣
↪1234
```

- Assign the token to a zone (it says KSK, but because there is no ZSK, this will become a CSK):

```
pdnsutil hsm assign example.com ecdsa256 ksk softhsm2 my-pkcs11-dnskey 1234
↪'my key' 'my pub key'
```

- Create the key (for 25, use the ID shown by the previous command):

```
pdnsutil hsm create-key example.com 25
```

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

```
pdnsutil show-zone example.com
```

9.8.2 SoftHSM2 with forwarding

Based on https://p11-glue.github.io/p11-glue/p11-kit/manual/remoting.html.

You need to install `gnutls-bin` to get token URLs.

You cannot run `p11-kit server` as root, so you will need some user for running it. This user must be in the `softhsm` group.

These commands need to be run as the non-root user (we shall call it `tokenuser`).

First, set up your token:

```
softhsm2-util --init-token --label "ecdsa#1" --pin 1234 --so-pin 1234 --free
pkcs11-tool --module /usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so --keypairgen␣
↪--key-type EC:prime256v1 --pin 1234 -a 'my key' --token-label "ecdsa#1"
```

Ensure it's there:

```
pkcs11-tool --module /usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so -l -O -p 1234
```

Get the URL for `p11-kit server`, which is needed for the server:

```
p11tool --provider /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-client.so --list-tokens
```

Set up forwarding:

```
cat <<EOF > /etc/pkcs11/modules/p11-kit-client.module
module: /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-client.so
EOF

p11-kit server -u pdns --provider /usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so
↪"pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;
↪serial=29fdc44dc0d61539;token=ecdsa%231"
P11_KIT_SERVER_ADDRESS=unix:path=/run/user/1000/p11-kit/pkcs11-5198; export P11_
↪KIT_SERVER_ADDRESS;
P11_KIT_SERVER_PID=5199; export P11_KIT_SERVER_PID;
```

You will need those values in PowerDNS running environment. Now you can verify that the token is reachable as `pdns` user with:

```
pkcs11-tool --module /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-client.so -T
Available slots:
Slot 0 (0x10): SoftHSM slot ID 0x40d61539
  token label        : ecdsa#1
  token manufacturer : SoftHSM project
  token model        : SoftHSM v2
  token flags        : login required, rng, token initialized, PIN initialized,␣
↪other flags=0x20
  hardware version   : 2.5
  firmware version   : 2.5
  serial num         : 29fdc44dc0d61539
  pin min/max        : 4/255
```

Then assign the HSM token to your zone with:

```
pdnsutil hsm assign example.com ecdsa256 ksk p11-kit-client 'ecdsa#1' 1234 'my key'
```

And then verify with `show-zone` that the zone now has a valid key.

You can do this over SSH as well (note that the example connects from token server to DNS server):

```
ssh -R /var/run/pdns/pkcs11:${P11_KIT_SERVER_ADDRESS#*=} pdns@server
export P11_KIT_SERVER_ADDRESS=/var/run/pdns/pkcs11
```

Verify that the token is visible:

```
pkcs11-tool --module /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-client.so -T
```

Then use the `pdnsutil hsm assign` command like before to assign the key to your zone; now you have DNSSEC over SSH.

9.8.3 Using CryptAS

Instructions on how to use CryptAS Athena IDProtect Key USB Token V2J Smart Card token on Ubuntu 14.04.

- 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`.
- Create file named `athena.module` with contents:

  ```
  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
  ```

(continues on next page)

(continued from previous page)

```
        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=/lib64/libASEP11.so -l -p some-pin -k --key-type RSA:2048␣
↪-a zone-ksk
pkcs11-tool --module=/lib64/libASEP11.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
  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 primary. It has been observed to produce about 1.5 signatures/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
- Chris Hofstaedtler
- Ruben d'Arco
- Morten Stevens
- Pieter Lexis

and everyone else who contributed to making this possible.

CHAPTER

TEN

PER ZONE SETTINGS: DOMAIN METADATA

Each served zone can have “metadata”. Such metadata determines how this zone behaves in certain circumstances.

> **Warning:** When multiple backends are in use, domain metadata is only retrieved from and written to the first DNSSEC-capable or metadata-capable backend, no matter where the related zones live.

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`.

The following options can only be read (not written to) via the HTTP API metadata endpoint.

- API-RECTIFY
- AXFR-MASTER-TSIG
- LUA-AXFR-SCRIPT
- NSEC3NARROW
- NSEC3PARAM
- PRESIGNED
- TSIG-ALLOW-AXFR

The option SOA-EDIT-API cannot be written or read via the HTTP API metadata endpoint.

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 secondaries in.

Example:

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

Each ACL has its own row in the database:

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

(continues on next page)

(continued from previous page)

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 value means that it will not be rectified. If this is not set at all, rectifying of the zone depends on the config variable *default-api-rectify*.

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 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
```

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

Changed in version 4.3.1: GSS support was removed

Changed in version 4.7.0: GSS support was added back

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 secondary 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 0 -".

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 records as 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*.

Global defaults for these values can be set via *default-publish-cdnskey* and *default-publish-cds*.

10.15 SLAVE-RENOTIFY

New in version 4.3.0.

If set to 1, will make PowerDNS renotify the secondaries after an AXFR is received from a master. Any other value means that no renotifies are done. If not set at all, action will depend on the *secondary-do-renotify* setting.

10.16 SOA-EDIT

When serving this zone, modify the SOA serial number in one of several ways. Mostly useful to get secondaries to re-transfer a zone regularly to get fresh RRSIGs. See the *DNSSEC documentation* for more information.

10.17 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`.

10.18 TSIG-ALLOW-AXFR

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

10.19 TSIG-ALLOW-DNSUPDATE

This setting allows you to set the TSIG key required to do an *Dynamic DNS Update (RFC 2136)*. If *GSS-TSIG* is enabled, you can put Kerberos principals here as well.

10.20 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-“.

CHAPTER

ELEVEN

DYNAMIC DNS UPDATE (RFC 2136)

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 SIG (TSIG and GSS*TSIG are 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*
- *gpgsql*
- *gsqlite3*
- *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 `dnsupdate-require-tsig`

New in version 5.0.0.

A setting to require DNS updates to be signed by a valid TSIG signature. The default is no, which means zones without TSIG keys can be updated by unauthenticated agents operating from an allowed address range.

11.1.4 `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.

11.1.5 `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 *above*). This will allow 198.51.100.0/8 and 203.0.113.2/32 to send DNS update messages for the example.org domain:

```
pdnsutil set-meta example.org ALLOW-DNSUPDATE-FROM 198.51.100.0/8 203.0.113.2/32
```

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. Here is an example using **`pdnsutil`** to create a key named *test*:

```
$ pdnsutil generate-tsig-key test hmac-sha512
Create new TSIG key test hmac-sha512 [base64-encoded key]

$ pdnsutil list-tsig-keys | grep test
test. hmac-sha512. [base64-encoded key]
```

This adds the key with the name *test* to the zone's metadata. Note, the keys need to be added separately with *add-meta*, not as a comma or space-separated list:

```
$ pdnsutil add-meta example.org TSIG-ALLOW-DNSUPDATE test
Set 'example.org' meta TSIG-ALLOW-DNSUPDATE = test

$ pdnsutil get-meta example.org TSIG-ALLOW-DNSUPDATE
TSIG-ALLOW-DNSUPDATE = test
```

This is an example of using the new *test* TSIG key with the **nsupdate** command (see the manpage for **nsupdate** for full details):

```
$ nsupdate <<!
server 127.0.0.1 53
zone example.org
update add test1.example.org 3600 A 1.2.3.4
update add test1.example.org 3600 TXT "this is a test"
key hmac-sha512:test [base64-encoded key]
send
!

$ dig +noall +answer -t any test1.example.org @127.0.0.1
test1.example.org.  3600    IN      A       1.2.3.4
test1.example.org.  3600    IN      TXT     "this is a test"
```

If any TSIG keys are listed in a zone's `TSIG-ALLOW-DNSUPDATE` metadata, one of them is required for updates. If `ALLOW-DNSUPDATE-FROM` is also set, both requirements need to be satisfied before an update will be accepted.

By default, an update can add, update or delete any resource records in the zone. See *Update policy* for finer-grained control of what an update is allowed to do. Use *dnsupdate-require-tsig* to disallow unsigned updates.

11.2.3 FORWARD-DNSUPDATE

See *Configuration options* for what it does, but per domain:

```
pdnsutil set-meta example.org FORWARD-DNSUPDATE 'yes'
```

The existence of the entry (even with an empty value) 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 than 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:

```
pdnsutil set-meta example.org 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 updated domain, might have an unexpected result due to a SOA-EDIT setting.

An example:

```
pdnsutil set-meta example.org 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. Exception: with SOA-EDIT=INCEPTION-EPOCH, the serial is bumped to at least the current EPOCH time.

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 FYhvwsW1ZtFZqWzsMpqhbg==
```

The important bits are the name of the key (**dhcpdupdate**) and the hash of the key (**FYhvwsW1ZtFZqWzsMpqhbg==**)

Using the details from the freshly generated key, add the following to your dhcpd.conf:

```
key "dhcpdupdate" {
        algorithm hmac-md5;
        secret "FYhvwsW1ZtFZqWzsMpqhbg==";
};
```

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 dhcpd 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 dhcpd.conf manual.

Per subnet, you also have to tell **dhcpd** 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 dhcpdupdate;
}

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

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

1. Which domain to use (**ddns-domainname “example.org”;**)
2. Which reverse-domain to use (**ddns-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 dhcpdupdate;**). We defined the key earlier.

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

11.4.2 Setting up PowerDNS

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

Enable 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.

```
pdnsutil set-meta example.org ALLOW-DNSUPDATE-FROM 127.0.0.1
pdnsutil set-meta 1.168.192.in-addr.arpa 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 **dhcpd**.

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

```
pdnsutil import-tsig-key dhcpdupdate hmac-md5 FYhvwsW1ZtFZqWzsMpqhbg==
pdnsutil set-meta example.org TSIG-ALLOW-DNSUPDATE dhcpdupdate
pdnsutil set-meta 1.168.192.in-addr.arpa TSIG-ALLOW-DNSUPDATE dhcpdupdate
```

This will:

1. Add the 'dhcpdupdate' 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 skip up to 7.
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, 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

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
  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
```

(continues on next page)

(continued from previous page)

```
    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
```

Additional updatepolicy example scripts can be found in our Wiki.

CHAPTER

TWELVE

CATALOG ZONES (RFC 9432)

Starting with the PowerDNS Authoritative Server 4.7.0, catalog zone support is available.

12.1 Supported catalog versions

Catalog version	Producer	Consumer
1 (ISC)	No	Yes
2 (**RFC 9432**)	Yes	Yes

All the important features of catalog zones version “2” are supported. There are however a few properties where support is limited:

- There is no support for group templates on consumers;
- There is no support for custom extensions;

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

- *gmysql*
- *gpgsql*
- *gsqlite3*
- *godbc*
- *lmdb*

12.2 Configuration options

None really.

12.3 Per zone settings

It is highly recommended to protect catalog zones with *TSIG*.

12.3.1 CATALOG-HASH

Producer zones store the member state as a hash in this metadata setting. This setting is managed by the authoritative server. Modifying or deleting this value will result in a serial increase of the producer zone and the update or recreation of this value.

12.4 Setting up catalog zones

Note: Catalog zone specification and operation is described in RFC 9432.

12.4.1 Setting up a producer zone

Setting up a producer zone is not very different from a regular primary zone. A producer zone is a minimal zone of type PRODUCER with only SOA and NS records at apex. All the records in a producer zone are ignored while generating a catalog.

An initial producer zone may look like this:

```
$TTL 3600
$ORIGIN catalog.example.
@               IN      SOA     invalid. hostmaster.invalid. (
                        1     ; serial
                        1H    ; refresh
                        10M   ; retry
                        1W    ; expire
                        1800  ; negTTL
                        )

@               IN      NS      invalid.
```

An interesting detail is the SOA serial: since the serial of a producer zone is automatically updated, it is important for the initial serial to be equal or lower than epoch. This serial is increased to EPOCH after each relevant member update.

Create a producer zone:

```
pdnsutil load-zone catalog.example ZONEFILE
pdnsutil set-kind catalog.example producer
```

Creating producer zones is supported in the *API*, using type `PRODUCER`.

12.4.2 Assigning members to a producer zone

After the producer zone is created it is necessary to assign member zones to it. In the example below `example.com` is the member and `catalog.example` is the catalog.

```
pdnsutil set-catalog example.com catalog.example
pdnsutil set-kind example.com primary
```

Setting catalog values is supported in the *API*, by setting the `catalog` property in the zone properties. Setting the catalog to an empty `""` removes the member zone from the catalog it is in.

Each member zone may have one or more additional properties as defined in the RFC. PowerDNS currently supports the following properties:

- coo - A single DNSName
- group - Multiple string values for group are allowed

```
pdnsutil set-option example.com producer coo other-catalog.example
pdnsutil set-option example.com producer group pdns-group-x pdns-group-y
```

There is also an option to set a specific <unique-N> value for a zone. This is done by setting a the `unique` value. This is used to signal a state reset to the consumer. The value for `unique` is a single DNS label.

```
pdnsutil --config-dir=. --config-name=gmysql set-option test.com producer unique␣
↪123
```

Setting options is not yet supported in the API.

12.4.3 Setting up a consumer zone

Setting up a consumer zone on a secondary server is almost identical to a normal secondary zone. The only difference is the type, which is now set to CONSUMER.

```
pdnsutil create-secondary-zone catalog.example 192.0.2.42
pdnsutil set-kind catalog.example consumer
```

Creating consumer zones is supported in the *API*, using type `CONSUMER`.

New member zones on the consumer adopt their primaries from the consumer zone.

12.4.4 Updating a secondary server when primary address/port changes

If NOTIFY+AXFR are being used to replicate zone changes between the primary and secondary servers, and the address and/or port of the primary server changes, two steps are necessary on each secondary server in order to fully apply the changes.

```
pdnsutil change-secondary-zone-primary catalog.example 192.0.2.45
pdns_control retrieve catalog.example
```

This will update the primary server contact details in each zone included in the catalog zone.

CHAPTER

THIRTEEN

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

13.1 Provisioning outbound AXFR access

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

Warning: Any host with the correct TSIG key will be able to perform the AXFR, even if the host is not within the defined `allow-axfr-ips` ranges.

Another way 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=";
```

(continues on next page)

(continued from previous page)

```
};

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

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

13.2 Provisioning signed notification and AXFR requests

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

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

For the 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.

13.3 GSS-TSIG support

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

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

13.3.1 Prerequisites

- Working Kerberos environment. Please refer to your Kerberos vendor documentation on how to set it up.
- Service Principal(s) (of the form `DNS/your.dns.server.name@REALM`) in either per-user keytab or system keytab, where `your.dns.server.name` must match the nameserver name in the SOA record of the zone. If a user keytab is used, specify it using the `KRB5_KTNAME` environment variable when starting up PDNS server, which must be able to read the keytab file.

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

13.3.2 Setting up

To allow AXFR / DNS update to work, you need to set *enable-gss-tsig* and configure `GSS-ACCEPTOR-PRINCIPAL` in *Per zone settings: Domain Metadata*. This will define the principal that is used to accept any GSS context requests for names in the specified domain. This *must* match to a principal in the keytab used by PDNS Server. 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 (client) principal names you intend to allow either AXFR or DNS update. No wildcards accepted. If a Lua update policy is defined (see *Dynamic DNS Update (RFC 2136)*) no `TSIG-ALLOW-DNSUPDATE` entries are needed, as the Lua policy defines which principals can update which records.

CHAPTER

FOURTEEN

LUA RECORDS

To facilitate dynamic behaviour, such as Global Server Load Balancing, PowerDNS Authoritative Server version 4.2 and later support dynamic DNS records.

These records contain small snippets of configuration that enable dynamic behaviour based on requestor IP address, requestor's EDNS Client Subnet, server availability or other factors.

Capabilities range from very simple to highly advanced multi-pool geographically & weighed load balanced IP address selection.

Although users need not be aware, PowerDNS dynamic DNS records are actually tiny (or larger) Lua statements.

Note: This is a PowerDNS specific feature, and is not (yet) standardized by the IETF or other standards bodies. We are committed however to interoperability, and strive to turn this functionality into a broadly supported standard.

To enable this feature, either set *enable-lua-records* in the configuration, or set the `ENABLE-LUA-RECORDS` per-zone metadata item to `1`.

In addition, to benefit from the geographical features, make sure the PowerDNS launch statement includes the `geoip` backend.

Warning: When using AXFR to transfer long LUA records, ensure that the record content is split into segments no longer than 255 bytes. Otherwise, due to the nature of the transfer method (as TXT content), the content will be split into 255 byte chunks. This can lead to the code being split in the wrong places on the secondary.

14.1 Examples

Before delving into the details, some examples may be of use to explain what dynamic records can do.

Here is a very basic example using *`ifportup()`*:

```
www    IN    LUA    A    "ifportup(443, {'192.0.2.1', '192.0.2.2'})"
```

This turns the 'www' name within a zone into a special record that will randomly return 192.0.2.1 or 192.0.2.2, as long as both of these IP addresses listen on port 443.

If either IP address stops listening, only the other address will be returned. If all IP addresses are down, all candidates are returned.

You can also provide multiple sets of IP addresses to prioritize a set over the rest. If an IP address from the first set is available, it will be returned. If no addresses work in the first set, the second set is tried.

For example:

```
www    IN    LUA    A    "ifportup(443, {{'192.0.2.1', '192.0.2.2'}, {'192.0.3.1'}}
↪)"
```

Because DNS queries require rapid answers, server availability is not checked synchronously. In the background, a process periodically determines if IP addresses mentioned in availability rules are, in fact, available.

Another example using `pickclosest()`:

```
www    IN    LUA    A    "pickclosest({'192.0.2.1','192.0.2.2','198.51.100.1'})"
```

This uses the GeoIP backend to find indications of the geographical location of the requestor and the listed IP addresses. It will return with one of the closest addresses.

`pickclosest()` and `ifportup()` can be combined as follows:

```
www    IN    LUA    A    ("ifportup(443, {'192.0.2.1', '192.0.2.2', '198.51.100.1'}
↪"
                          ", {selector='pickclosest'})
↪")
```

This will pick from the viable IP addresses the one deemed closest to the user.

LUA records can also contain more complex code, for example:

```
www    IN    LUA    A    ";if country('US') then return {'192.0.2.1','192.0.2.2',
↪'198.51.100.1'} else return '192.0.2.2' end"
```

As you can see you can return both single string value or array of strings.

An example Lua record accessing `qname`:

```
*.example.net   10      IN      LUA     TXT "; return 'Got a TXT query for ' ..␣
↪qname:toString() .. '; First label is: ' .. qname:getRawLabels()[1]"
```

`qtype` cannot be accessed from a Lua script, the value is fixed per Lua record. See *Preset variables* for available variables.

14.2 Using LUA Records with Generic SQL backends

It's possible to use Lua records with the Generic SQL backends such as gmysql and gpgsql.

Be aware that due to the fact that Lua records uses both double and single quotes, you will need to appropriately escape them in INSERT/UPDATE queries.

Here is an example from the previous section (`pickclosest`) which should work for both **MySQL** and **PostgreSQL**:

```
-- Create the zone example.com
INSERT INTO domains (id, name, type) VALUES (1, 'example.com', 'NATIVE');

-- Enable Lua records for the zone (if not enabled globally)
INSERT INTO domainmetadata (domain_id, kind, content)
VALUES (1, 'ENABLE-LUA-RECORDS', 1);

-- Create a pickClosest() Lua A record.
-- Double single quotes are used to escape single quotes in both MySQL and␣
↪PostgreSQL
INSERT INTO records (domain_id, name, type, content, ttl)
VALUES (
  1,
  'www.example.com',
```

(continues on next page)

(continued from previous page)

```
  'LUA',
  'A "pickclosest({''192.0.2.1'',''192.0.2.2'',''198.51.100.1''})"',
  600
);
```

The above queries create a zone `example.com`, enable Lua records for the zone using `ENABLE-LUA-RECORDS`, and finally insert a LUA A record for the `www` subdomain using the previous pickclosest example.

See *Details & Security* for more information about enabling Lua records, and the risks involved.

14.3 Record format

Note: The fine authors of the Lua programming language insist that it is Lua and not LUA. Lua means 'moon' in Portuguese, and it is not an abbreviation. Sadly, it is DNS convention for record types to be all uppercase. Sorry.

The LUA record consists of an initial query type, which is the selector on which the snippet will trigger. Optionally this query type itself can be LUA again for configuration scripts. The query type is then followed by the actual Lua snippet.

LUA records can have TTL settings, and these will be honoured. In addition, LUA records output can be DNSSEC signed like any other record, but see below for further details.

14.4 More powerful example

A more powerful example:

```
west    IN    LUA    A    ( "ifurlup('https://www.lua.org/',                  "
                            "{{'192.0.2.1', '192.0.2.2'}, {'198.51.100.1'}},  "
                            "{stringmatch='Programming in Lua'})              " )
```

In this case, IP addresses are tested to see if they will serve https for 'www.lua.org', and if that page contains the string `Programming in Lua`.

Two sets of IP addresses are supplied. If an IP address from the first set is available, it will be returned. If no addresses work in the first set, the second set is tried.

This configuration makes sense in the following context:

```
www    IN    LUA    CNAME   ( ";if(continent('EU')) then return 'west.powerdns.org
↪' "
                              "else return 'usa.powerdns.org' end" )
```

This sends queries that are geolocated to Europe to 'west.powerdns.org', and the rest to 'usa.powerdns.org'. The configuration for that name would then be:

```
usa    IN    LUA    A    ( "ifurlup('https://www.lua.org/',           "
                           "{{'198.51.100.1'}, {'192.0.2.1', '192.0.2.2'}},  "
                           "{stringmatch='Programming in Lua'})              " )
```

Note that the sets of IP addresses have reversed order - visitors geolocated outside of Europe will hit 198.51.100.1 as long as it is available, and the 192.0.2.1 and 192.0.2.2 servers as backup.

14.5 Advanced topics

By default, LUA records are executed as if they were the argument to Lua's `return` statement. This saves a lot of typing for common cases. To run actual Lua scripts, start a record with a semicolon (`;`). You need to add your own `return` statement.

To keep records more concise and readable, configuration can be stored in separate records. The full example from above can also be written as:

```
config    IN    LUA    LUA ("settings={stringmatch='Programming in Lua'}  "
                            "EUips={'192.0.2.1', '192.0.2.2'}             "
                            "USAips={'198.51.100.1'}                      ")

www       IN    LUA    CNAME ( ";if(continent('EU')) then return 'west.powerdns.org
↪' "
                               "else return 'usa.powerdns.org' end" )

usa       IN    LUA    A     ( ";include('config')                              "
                               "return ifurlup('https://www.lua.org/',       "
                               "{USAips, EUips}, settings)                   " )

west      IN    LUA    A     ( ";include('config')                              "
                               "return ifurlup('https://www.lua.org/',       "
                               "{EUips, USAips}, settings)                   " )
```

14.6 Details & Security

LUA records are synthesized on query. They can also be transferred via AXFR to other PowerDNS servers.

LUA records themselves cannot be queried however, as this would allow third parties to see load balancing internals they do not need to see.

A non-supporting DNS server will also serve a zone with LUA records, but they will not function, and will in fact leak the content of the LUA records.

Note: Under NO circumstances serve LUA records from zones from untrusted sources! LUA records will be able to bring down your system and possible take over control of it. Use TSIG on AXFR even from trusted sources!

LUA records can be DNSSEC signed, but because they are dynamic, it is not possible to combine pre-signed DNSSEC zone and LUA records. In other words, the signing key must be available on the server creating answers based on LUA records.

Note that to protect operators, support for LUA records must be enabled explicitly, either globally (`enable-lua-records`) or per zone (`ENABLE-LUA-RECORDS` = 1).

14.7 Shared Lua state model

The default mode of operation for LUA records is to create a fresh Lua state for every query that hits a LUA record. This way, different LUA records cannot accidentally interfere with each other, by leaving around global objects, or perhaps even deleting relevant functions. However, creating a Lua state (and registering all our functions for it, see Reference below) takes measurable time. For users that are confident they can write Lua scripts that will not interfere with each other, a mode is supported where Lua states are created on the first query, and then reused forever. Note that the state is per-thread (for UDP, plus one shared state for all TCP), so while data sharing between LUA invocations is possible (useful for caching and reducing the cost of `require`), there is no single shared Lua environment. In non-scientific testing this has yielded up to 10x QPS increases.

To use this mode, set `enable-lua-records=shared`. Note that this enables LUA records for all zones.

14.8 Reference

14.8.1 Preset variables

LUA rules run within the same environment as described in *DNS Modes of Operation*.

The Lua snippets can query the following variables:

Query variables

dh The *DNSHeader* of the received query.

dnssecOK A boolean describing if the DNSSEC OK (DO) bit was set in the query.

ednsPKTSize The advertised EDNS buffer size.

qname The name of the requested record. This is a *DNSName*.

zone The zone this LUA record is in. This is a *DNSName*.

zoneid The id of the zone. This is an integer.

tcp Whether or not the query was received over TCP.

Client variables

ecswho The EDNS Client Subnet, should one have been set on the query. Unset otherwise. This is a *Netmask*.

bestwho In absence of ECS, this is set to the IP address of requesting resolver. Otherwise set to the network part of the EDNS Client Subnet supplied by the resolver. This is a *ComboAddress*.

who IP address of requesting resolver as a *ComboAddress*.

localwho IP address (including port) of socket on which the question arrived.

14.8.2 Functions available

Record creation functions

ifportup (*portnum*, *addresses*[, *options*])

Simplistic test to see if an IP address listens on a certain port. This will attempt a TCP connection on port `portnum` and consider it available if the connection establishes. No data will be sent or read on that connection. Note that both IPv4 and IPv6 addresses can be tested, but that it is an error to list IPv4 addresses on an AAAA record, or IPv6 addresses on an A record.

Will return a single address from the set of available addresses. If no address is available, will return a random element of the set of addresses supplied for testing.

Parameters

- **portnum** (*int*) – The port number to test connections to.
- **addresses** (*{str}*) – The list of addresses to check connectivity for.
- **options** – Table of options for this specific check, see below.

Various options can be set in the `options` parameter:

- `selector`: used to pick the address(es) from the list of available addresses. Choices include 'pickclosest', 'random', 'hashed', 'all' (default 'random').
- `backupSelector`: used to pick the address(es) from all addresses if all addresses are down. Choices include 'pickclosest', 'random', 'hashed', 'all' (default 'random').

- `source`: Source address to check from
- `timeout`: Maximum time in seconds that you allow the check to take (default 2)
- `interval`: Time interval between two checks, in seconds. Defaults to *lua-health-checks-interval* if not specified.
- `minimumFailures`: The number of unsuccessful checks in a row required to mark the address as down. Defaults to 1 if not specified, i.e. report as down on the first unsuccessful check.

ifurlup(*url*, *addresses*[, *options*])

More sophisticated test that attempts an actual http(s) connection to `url`. In addition, a list of sets of IP addresses can be supplied. The first set with at least one available address is selected. The `selector` then selects from the subset of available addresses of the selected set. An URL is considered available if the HTTP response code is 200 and optionally if the content matches the `stringmatch` option.

Parameters

- **url** (*string*) – The url to retrieve.
- **addresses** – List of sets of addresses to check the URL on.
- **options** – Table of options for this specific check, see below.

Various options can be set in the `options` parameter:

- `selector`: used to pick the address(es) from the subset of available addresses of the selected set. Choices include 'pickclosest', 'random', 'hashed', 'all' (default 'random').
- `backupSelector`: used to pick the address from all addresses if all addresses are down. Choices include 'pickclosest', 'random', 'hashed', 'all' (default 'random').
- `source`: Source address to check from
- `timeout`: Maximum time in seconds that you allow the check to take (default 2)
- `interval`: Time interval between two checks, in seconds. Defaults to *lua-health-checks-interval* if not specified.
- `stringmatch`: check `url` for this string, only declare 'up' if found
- `useragent`: Set the HTTP "User-Agent" header in the requests. By default it is set to "PowerDNS Authoritative Server"
- `byteslimit`: Limit the maximum download size to `byteslimit` bytes (default 0 meaning no limit).
- `minimumFailures`: The number of unsuccessful checks in a row required to mark the address as down. Defaults to 1 if not specified, i.e. report as down on the first unsuccessful check.

An example of a list of address sets:

```
ifurlup("https://example.com/", { {"192.0.2.20", "203.0.113.4"}, {"203.0.113.2
↪"} })
```

ifurlextup(*groups-of-address-url-pairs*[, *options*])

Very similar to `ifurlup`, but the returned IPs are decoupled from their external health check URLs. This is useful when health checking already happens elsewhere, and that state is exposed over HTTP(S). Health checks are considered positive if the HTTP response code is 200 and optionally if the content matches the `stringmatch` option.

Options are identical to those for `ifurlup`.

Example:

```
ifurlextup({{['192.168.0.1']='https://example.com/',['192.168.0.2']='https://
↪example.com/404'}})
```

Example with two groups:

```
ifurlextup({{['192.168.0.1']='https://example.net/404',['192.168.0.2']='https:/
↪/example.com/404'}, {['192.168.0.3']='https://example.net/'}})"
```

The health checker will look up the first two URLs (using normal DNS resolution to find them - whenever possible, use URLs with IPs in them). The 404s will cause the first group of IPs to get marked as down, after which the URL in the second group is tested. The third IP will get marked up assuming `https://example.net/` responds with HTTP response code 200.

pickrandom(*values*)

Returns a random value from the list supplied.

Parameters **values** – A list of strings such as IPv4 or IPv6 address.

This function also works for CNAME or TXT records.

pickselfweighted(*url*, *addresses*[, *options*])

Selects an IP address from the supplied list, weighted according to the results of *isUp* checks. Each address is evaluated, and if its associated weight (from *isUp*) is greater than 0, it is considered for selection using a weighted hash based on *bestwho*. If no address is “up” the function defaults to a random selection.

Parameters

- **url** (*string*) – The health check url to retrieve.
- **addresses** – A list of IP addresses to evaluate.
- **options** – Table of options for this specific check, see below.

Various options can be set in the `options` parameter:

- `selector`: used to pick the address(es) from the subset of available addresses of the selected set. Choices include ‘pickclosest’, ‘random’, ‘hashed’, ‘all’ (default ‘random’).
- `backupSelector`: used to pick the address from all addresses if all addresses are down. Choices include ‘pickclosest’, ‘random’, ‘hashed’, ‘all’ (default ‘random’).
- `source`: Source address to check from
- `timeout`: Maximum time in seconds that you allow the check to take (default 2)
- `stringmatch`: check `url` for this string, only declare ‘up’ if found
- `useragent`: Set the HTTP “User-Agent” header in the requests. By default it is set to “PowerDNS Authoritative Server”
- `byteslimit`: Limit the maximum download size to `byteslimit` bytes (default 0 meaning no limit).

An example of a list of address sets:

```
pickselfweighted("http://example.com/weight", { "192.0.2.20", "203.0.113.4",
↪"203.0.113.2" })
```

This function is ideal for scenarios where candidates can self-determine their weights, while also providing fallback behavior when all addresses are down.

pickrandomsample(*number*, *values*)

Returns N random values from the list supplied.

Parameters

- **number** – Number of values to return
- **values** – A list of strings such as IPv4 or IPv6 address.

This function also works for CNAME or TXT records.

pickhashed(*values*)

Based on the hash of `bestwho`, returns a random value from the list supplied.

Parameters values – A list of strings such as IPv4 or IPv6 address.

This function also works for CNAME or TXT records.

pickclosest(*addresses*)
Returns IP address deemed closest to the `bestwho` IP address.

Parameters addresses – A list of strings with the possible IP addresses.

latlon()
Returns text listing fractional latitude/longitude associated with the `bestwho` IP address.

latlonloc()
Returns text in LOC record format listing latitude/longitude associated with the `bestwho` IP address.

closestMagic()
Suitable for use as a wildcard LUA A record. Will parse the query name which should be in format:

```
192-0-2-1.192-0-2-2.198-51-100-1.magic.v4.powerdns.org
```

It will then resolve to an A record with the IP address closest to `bestwho` from the list of supplied addresses.

In the `magic.v4.powerdns.org` this looks like:

```
*.magic.v4.powerdns.org    IN    LUA    A    "closestMagic()"
```

In another zone, a record is then present like this:

```
www-balanced.powerdns.org    IN    CNAME    192-0-2-1.192-0-2-2.198-51-100-1.
↪magic.v4.powerdns.org
```

This effectively opens up your server to being a 'geographical load balancer as a service'.

Performs no uptime checking.

all(*values*)
Returns all values.

Parameters values – A list of strings such as IPv4 or IPv6 address.

This function also works for CNAME or TXT records.

view(*pairs*)
Shorthand function to implement 'views' for all record types.

Parameters pairs – A list of netmask/result pairs.

An example:

```
view.v4.powerdns.org    IN    LUA    A ("view({                              ␣
↪   "
                                        "{ {'192.168.0.0/16'}, {'192.168.1.54'}
↪},"
                                        "{ {'0.0.0.0/0'}, {'192.0.2.1'}}       ␣
↪   "
                                        " }) " )
```

This will return IP address 192.168.1.54 for queries coming from 192.168.0.0/16, and 192.0.2.1 for all other queries.

This function also works for CNAME or TXT records.

pickchashed(*values*)
Based on the hash of `bestwho`, returns a string from the list supplied, as weighted by the various `weight` parameters and distributed consistently. Performs no uptime checking.

Parameters values – table of weight, string (such as IPv4 or IPv6 address).

This function works almost like `pickwhashed()` while bringing the following properties:

- reordering the list of entries won't affect the distribution
- updating the weight of an entry will only affect a part of the distribution
- because of the previous properties, the CPU and memory cost is a bit higher than `pickwhashed()`

Hashes will be pre computed the first time such a record is hit and refreshed if needed. If updating the list is done often, the cache may grow. A cleanup routine is performed every *lua-consistent-hashes-cleanup-interval* seconds (default 1h) and cleans cached entries for records that haven't been used for *lua-consistent-hashes-expire-delay* seconds (default 24h)

An example:

```
mydomain.example.com    IN    LUA    A ("pickchashed({                                 "
                                       "        {15,  "192.0.2.1"},                       "
                                       "        {100, "198.51.100.5"}                     "
                                       "})                                                ")
```

pickwhashed(*values*)

Based on the hash of `bestwho`, returns a string from the list supplied, as weighted by the various `weight` parameters. Performs no uptime checking.

Parameters **values** – table of weight, string (such as IPv4 or IPv6 address).

Because of the hash, the same client keeps getting the same answer, but given sufficient clients, the load is still spread according to the weight factors.

This function also works for CNAME or TXT records.

An example:

```
mydomain.example.com    IN    LUA    A ("pickwhashed({                                 "
                                       "        {15,  "192.0.2.1"},                       "
                                       "        {100, "198.51.100.5"}                     "
                                       "})                                                ")
```

picknamehashed(*values*)

Based on the hash of the DNS record name, returns a string from the list supplied, as weighted by the various `weight` parameters. Performs no uptime checking.

Parameters **values** – table of weight, string (such as IPv4 or IPv6 address).

This allows basic persistent load balancing across a number of backends. It means that `test.mydomain.example.com` will always resolve to the same IP, but `test2.mydomain.example.com` may go elsewhere. This function is only useful for wildcard records.

This works similar to round-robin load balancing, but has the advantage of making traffic for the same domain always end up on the same server which can help cache hit rates.

This function also works for CNAME or TXT records.

An example:

```
*.mydomain.example.com    IN    LUA    A ("picknamehashed({                              "
                                         "        {15,  "192.0.2.1"},                       "
```

(continues on next page)

(continued from previous page)

```
                                        "          {100, "198.51.100.5"}        ␣
↪     "
                                        "})                                     ␣
↪     ")
```

pickwrandom(*values*)

Returns a random string from the list supplied, as weighted by the various `weight` parameters. Performs no uptime checking.

Parameters **values** – table of weight, string (such as IPv4 or IPv6 address).

See *pickwhashed()* for an example.

This function also works for CNAME or TXT records.

Reverse DNS functions

> **Warning:** For *createForward()* and *createForward6()*, we recommend filtering with *filterForward()*, to prevent PowerDNS from generating A/AAAA responses to addresses outside of your network. Not limiting responses like this may, in some situations, help attackers with impersonation and attacks like such as cookie stealing.

createReverse(*format*[, *exceptions*])

Used for generating default hostnames from IPv4 wildcard reverse DNS records, e.g. `*.0.0.127.in-addr.arpa`

See *createReverse6()* for IPv6 records (ip6.arpa)

See *createForward()* for creating the A records on a wildcard record such as `*.static.example.com`

Returns a formatted hostname based on the format string passed.

Parameters

- **format** – A hostname string to format, for example `%1%.%2%.%3%.%4%.static.example.com`.
- **exceptions** – An optional table of overrides. For example `{['10.10.10.10'] = 'quad10.example.com.'}` would, when generating a name for IP `10.10.10.10`, return `quad10.example.com` instead of something like `10.10.10.10.example.com`.

Formatting options:

- **`%1%` to `%4%` are individual octets**
  - Example record query: `1.0.0.127.in-addr.arpa`
  - `%1%` = 127
  - `%2%` = 0
  - `%3%` = 0
  - `%4%` = 1
- **`%5%` joins the four decimal octets together with dashes**
  - Example: `%5%.static.example.com` is equivalent to `%1%-%2%-%3%-%4%.static.example.com`
- **`%6%` converts each octet from decimal to hexadecimal and joins them together**
  - Example: A query for `15.0.0.127.in-addr.arpa`

  - `%6` would be `7f00000f` (127 is 7f, and 15 is 0f in hexadecimal)

Example records:

```
*.0.0.127.in-addr.arpa IN    LUA    PTR "createReverse('%1%.%2%.%3%.%4%.static.
↪example.com')"
*.1.0.127.in-addr.arpa IN    LUA    PTR "createReverse('%5%.static.example.com
↪')"
*.2.0.127.in-addr.arpa IN    LUA    PTR "createReverse('%6%.static.example.com
↪')"
```

When queried:

```
# -x is syntactic sugar to request the PTR record for an IPv4/v6 address such␣
↪as 127.0.0.5
# Equivalent to dig PTR 5.0.0.127.in-addr.arpa
$ dig +short -x 127.0.0.5 @ns1.example.com
127.0.0.5.static.example.com.
$ dig +short -x 127.0.1.5 @ns1.example.com
127-0-0-5.static.example.com.
$ dig +short -x 127.0.2.5 @ns1.example.com
7f000205.static.example.com.
```

`createForward``()`

Used to generate the reverse DNS domains made from *`createReverse()`*

Generates an A record for a dotted or hexadecimal IPv4 domain (e.g. 127.0.0.1.static.example.com)

It does not take any parameters, it simply interprets the zone record to find the IP address.

An example record for zone `static.example.com`:

```
*.static.example.com    IN    LUA    A "createForward()"
```

This function supports the forward dotted format (`127.0.0.1.static.example.com`), and the hex format, when prefixed by two ignored characters (`ip40414243.static.example.com`)

When queried:

```
$ dig +short A 127.0.0.5.static.example.com @ns1.example.com
127.0.0.5
```

Since 4.8.0: the hex format can be prefixed by any number of characters (within DNS label length limits), including zero characters (so no prefix).

`createReverse6``(`*format*[, *exceptions*]`)`

Used for generating default hostnames from IPv6 wildcard reverse DNS records, e.g. `*.1.0.0.2.ip6.arpa`

For simplicity purposes, only small sections of IPv6 rDNS domains are used in most parts of this guide, as a full ip6.arpa record is around 80 characters long.

See *`createReverse()`* for IPv4 records (in-addr.arpa)

See *`createForward6()`* for creating the AAAA records on a wildcard record such as `*.static.example.com`

Returns a formatted hostname based on the format string passed.

Parameters

- **`format`** – A hostname string to format, for example `%33%.static6.example.com`.
- **`exceptions`** – An optional table of overrides. For example `{['2001:db8::1'] = 'example.example.com.'}` would, when generating a name for IP

2001:db8::1, return example.example.com instead of something like 2001--db8.example.com.

Formatting options:

- **%1% to %32% are individual characters (nibbles)**
  - **Example PTR record query:** a.0.0.0.1.0.0.2.ip6.arpa
  - %1% = 2
  - %2% = 0
  - %3% = 0
  - %4% = 1
- %33% converts the compressed address format into a dashed format, e.g. 2001:a::1 to 2001-a--1
- **%34% to %41% represent the 8 uncompressed 2-byte chunks**
  - **Example:** PTR query for 2001:a:b::123
  - %34% - returns 2001 (chunk 1)
  - %35% - returns 000a (chunk 2)
  - %41% - returns 0123 (chunk 8)

Example records:

```
*.1.0.0.2.ip6.arpa IN    LUA    PTR "createReverse6('%33%.static6.example.com')
↪"
*.2.0.0.2.ip6.arpa IN    LUA    PTR "createReverse6('%34%.%35%.static6.example.
↪com')"
```

When queried:

```
# -x is syntactic sugar to request the PTR record for an IPv4/v6 address such␣
↪as 2001::1
# Equivalent to dig PTR 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.b.0.0.0.a.0.0.
↪0.1.0.0.2.ip6.arpa
# readable version:     1.0.0.0 .0.0.0.0 .0.0.0.0 .0.0.0.0 .0.0.0.0 .b.0.0.0 .
↪a.0.0.0 .1.0.0.2 .ip6.arpa

$ dig +short -x 2001:a:b::1 @ns1.example.com
2001-a-b--1.static6.example.com.

$ dig +short -x 2002:a:b::1 @ns1.example.com
2002.000a.static6.example.com
```

createForward6()

Used to generate the reverse DNS domains made from *createReverse6()*

Generates an AAAA record for a dashed compressed IPv6 domain (e.g. 2001-a-b--1.static6.example.com)

It does not take any parameters, it simply interprets the zone record to find the IP address.

An example record for zone static.example.com:

```
*.static6.example.com    IN    LUA    AAAA "createForward6()"
```

This function supports the dashed compressed format (i.e. 2001-a-b--1.static6.example.com), and the dot-split uncompressed format (2001.db8.6.5.4.3.2.1.static6.example.com)

When queried:

```
$ dig +short AAAA 2001-a-b--1.static6.example.com @ns1.example.com
2001:a:b::1
```

Since 4.8.0: a non-split full length format (`20010002000300040005000600070db8.example.com`) is also supported, optionally prefixed, in which case only the last 32 characters will be considered.

filterForward(*address*, *masks*[, *fallback*])

New in version 4.5.0.

Used for limiting the output of *createForward()* and *createForward6()* to a set of netmasks.

Parameters

- **address** – A string containing an address, usually taken directly from `createForward:  or :func:`createForward6().`
- **masks** – A NetmaskGroup; any address not matching the NMG will be replaced by the fallback address.
- **fallback** – A string containing the fallback address. Defaults to `0.0.0.0` or `::`.

Example:

```
*.static4.example.com IN LUA A "filterForward(createForward(), newNMG({'192.0.
↪2.0/24', '10.0.0.0/8'}))"
```

Since 4.9.0: if the fallback parameter is an empty string, `filterForward` returns an empty set, yielding a NODATA answer. You cannot combine this feature with DNSSEC.

Helper functions

asnum(*number*)
asnum(*numbers*)

Returns true if the `bestwho` IP address is determined to be from any of the listed AS numbers.

Parameters

- **number** (*int*) – An AS number
- **numbers** (*[int]*) – A list of AS numbers

country(*country*)
country(*countries*)

Returns true if the `bestwho` IP address of the client is within the two letter ISO country code passed, as described in *GeoIP backend*.

Parameters

- **country** (*string*) – A country code like "NL"
- **countries** (*[string]*) – A list of country codes

countryCode()

Returns two letter ISO country code based `bestwho` IP address, as described in *GeoIP backend*. If the two letter ISO country code is unknown "–" will be returned.

region(*region*)
region(*regions*)

Returns true if the `bestwho` IP address of the client is within the two letter ISO region code passed, as described in *GeoIP backend*.

Parameters

- **region** (*string*) – A region code like "CA"
- **regions** (*[string]*) – A list of regions codes

regionCode()

Returns two letter ISO region code based `bestwho` IP address, as described in *GeoIP backend*. If the two letter ISO region code is unknown "–" will be returned.

continent(*continent*)
continent(*continents*)

Returns true if the `bestwho` IP address of the client is within the continent passed, as described in *GeoIP backend*.

Parameters

- **continent** (*string*) – A continent code like "EU"
- **continents** (*[string]*) – A list of continent codes

continentCode()

Returns two letter ISO continent code based `bestwho` IP address, as described in *GeoIP backend*. If the two letter ISO continent code is unknown "–" will be returned.

netmask(*netmasks*)

Returns true if `bestwho` is within any of the listed subnets.

Parameters **netmasks** (*[string]*) – The list of IP addresses to check against

dblookup(*name*, *type*)

New in version 4.9.0.

Does a database lookup for name and type, and returns a (possibly empty) array of string results.

Please keep the following in mind:

- it does not evaluate any LUA code found
- if you needed just one string, perhaps you want `dblookup('www.example.org', pdns.A)[1]` to take the first item from the array
- some things, like ifurlup, don't like empty tables, so be careful not to accidentally look up a name that does not have any records of that type, if you are going to use the result in `ifurlup`

Example usage:

```
www IN LUA A "ifurlup('https://www.example.com/', {dblookup('www1.example.com', pdns.A), dblookup('www2.example.com', pdns.A), dblookup('www3.example.com', pdns.A)})"
```

Parameters

- **name** (*string*) – Name to look up in the database
- **type** (*int*) – DNS type to look for

14.8.3 LUA Reference

ComboAddress objects

IP addresses are moved around in a native format, called a ComboAddress. ComboAddresses can be IPv4 or IPv6, and unless you want to know, you don't need to.

Functions and methods of a `ComboAddress`

newCA(*address*) → ComboAddress

Returns a new *`ComboAddress`* object based on address

Parameters **address** (*string*) – The IP address, with optional port, to represent

```
addr = newCA("1.2.3.4")
```

newCAFromRaw(*rawaddress*[, *port*]) → ComboAddress

Returns a new *ComboAddress* object based on the 4- or 16-octet string. For example, `newCAFromRaw('ABCD')` makes a `ComboAddress` object holding the IP `65.66.67.68`, because those are the ASCII values for those four letters.

Parameters

- **rawaddress** (*string*) – The IPv4 of IPv6 address as a 4/16 octet string
- **port** (*int*) – The optional port number

class ComboAddress

A `ComboAddress` represents an IP address with possibly a port number. The object can be an IPv4 or an IPv6 address. It has these methods:

ComboAddress:getPort() -> int

Returns the port number.

ComboAddress:isIPv4() -> bool

Returns true if the address is an IPv4, false otherwise

ComboAddress:isIPv6() -> bool

Returns true if the address is an IPv6, false otherwise

ComboAddress:isMappedIPv4() -> bool

Returns true if the address is an IPv4 mapped into an IPv6, false otherwise

ComboAddress:mapToIPv4() -> ComboAddress

Convert an IPv4 address mapped in a v6 one into an IPv4. Returns a new ComboAddress

ComboAddress:toString() -> string

Returns in human-friendly format

ComboAddress:getRaw() -> string

Returns in raw bytes format format

ComboAddress:toStringWithPort() -> string

Returns in human-friendly format, with port number

ComboAddress:truncate(bits)

Truncate the ComboAddress to the specified number of bits. This essentially zeroes all bits after `bits`.

Parameters bits (*int*) – Amount of bits to truncate to

ComboAddressSet objects

We provide a convenient object class that can store unique ComboAddresses in no particular order and allows fast retrieval of individual elements based on their values

```
addr = newCA("1.2.3.4")
myset = newCAS()
myset:add(addr)
if myset:check(addr) then -- prints "found!"
  print('found!')
end
```

Functions and methods of a `ComboAddressSet`

newCAS() → ComboAddressSet

Returns an empty *ComboAddressSet* object

class ComboAddressSet

A `ComboAddressSet` can store multiple *ComboAddress*

It has these methods:

ComboAddressSet:add(addr)

ComboAddressSet:add(addrs)

ComboAddressSet:add(ca)

Add the given addresses to set. the parameter can be of the following types:

Parameters

- **addr** (*string*) – Handy way to add *ComboAddress* from its string representation
- **addrs** (*[string]*) – Add the given list of addresses to the set
- **ca** (`ComboAddress`) – The *ComboAddress* object to add to set

```
addr = newCA("1.2.3.4")
myset = newCAS()
myset:add(addr)
myset:add("5.6.7.8")
myset:add({"::1/128", "10.11.12.13"})
```

DNSHeader (dh) object

class DNSHeader

This object holds a representation of a DNS packet's header.

DNSHeader:getRD() -> bool

Get recursion desired flag.

DNSHeader:getCD() -> bool

Get checking disabled flag.

DNSHeader:getID() -> bool

Get header's ID

DNSHeader:getTC() -> bool

Truncated message bit

DNSHeader:getRA() -> bool

Recursion available

DNSHeader:getAD() -> bool

Authenticated data from name server

DNSHeader:getAA() -> bool

Authoritative answer

DNSHeader:getRCODE() -> int

Response code

DNSHeader:getOPCODE() -> int

Purpose of message

DNSHeader:getQDCOUNT() -> int

Number of question entries

DNSHeader:getANCOUNT() -> int

Number of answer entries

DNSHeader:getNSCOUNT() -> int

Number of authority entries

DNSHeader:getARCOUNT() -> int
Number of resource entries

DNSName objects

A *DNSName* object represents a name in the DNS. It has several functions that can manipulate it without conversions to strings. Creating a `DNSName` is done with the *newDN()*:

```
myname = newDN("www.example.com")
```

PowerDNS will complain loudly if the name is invalid (e.g. too long, dot in the wrong place).

The `myname` variable has several functions to get information from it

```
print(myname:countLabels()) -- prints "3"
print(myname:wireLength()) -- prints "17"
name2 = newDN("example.com")
if myname:isPartOf(name2) then -- prints "it is"
  print('it is')
end
```

Functions and methods of a DNSName

newDN(*name*) → DNSName
Returns the *DNSName* object of `name`.

Parameters **name** (*string*) – The name to create a DNSName for

class DNSName
A `DNSName` object represents a name in the DNS. It is returned by several functions and has several functions to programmatically interact with it.

DNSName:canonCompare(name) -> bool
Performs a comparison of DNS names in canonical order. Returns true if the DNSName comes before `name`. See https://tools.ietf.org/html/rfc4034#section-6

Parameters **name** (DNSName) – The name to compare to

DNSName:makeRelative(name) -> DNSName
Returns a new DNSName that is relative to `name`

```
name = newDN("bb.a.example.com.")
parent = newDN("example.com.")
rel = name:makeRelative(parent) -- contains DNSName("bb.a.")
```

Parameters **name** (DNSName) – The name to compare to

DNSName:isPartOf(name) -> bool
Returns true if the DNSName is part of the DNS tree of `name`.

Parameters **name** (DNSName) – The name to check against

DNSName:toString() -> string
Returns a human-readable form of the DNSName

DNSName:toStringNoDot() -> string
Returns a human-readable form of the DNSName without the trailing dot

DNSName:chopOff() -> bool
Removes the left-most label and returns `true`. `false` is returned if no label was removed

DNSName:countLabels() -> int
Returns the number of DNSLabels in the name

DNSName:wireLength() -> int
Returns the length in bytes of the DNSName as it would be on the wire.

DNSName::getRawLabels() -> [string]
Returns a table that contains the raw labels of the DNSName

DNSName::countLabels() -> int
Returns the number of labels of the DNSName

DNSName::equal(name) -> bool
Perform a comparison of the DNSName to the given `name`. You can also compare directly two DNSName objects using the == operator

Parameters **name** (*string*) – The name to compare to

DNSResourceRecord objects

A *DNSResourceRecord* object represents a resource record in the DNS. Creating a `DNSResourceRecord` is done with the *newDRR()*.

Functions and methods of a `DNSResourceRecord`

newDRR(*name*, *type*, *ttl*, *content*[, *domainId*[, *auth*]]) → DNSResourceRecord
Returns a new *DNSResourceRecord* object.

Parameters

- **name** (DNSName) – The name to the new record
- **type** (*string*) – The name to create a DNSName for
- **ttl** (*int*) – The TTL of the record
- **content** (*string*) – The content of the record
- **domainId** (*int*) – The optional domain ID of the zone the record belongs to
- **auth** (*int*) – Whether the record is authoritative

```
name = newDN("www.example.org.")
rr = new DRR(name, "IN", 3600, )
```

class DNSResourceRecord
A `DNSResourceRecord` object represents a DNS record.

DNSResourceRecord:toString() -> string
Returns the full content of the record as a string

DNSResourceRecord:qname() -> DNSName
Returns the name of the record

DNSResourceRecord:wildcardName() -> DNSName
Returns the wildcard name of the record the record was matched against

DNSResourceRecord:content() -> string
Returns what the record points to

DNSResourceRecord:lastModified() -> int
If non-zero, last time this record was changed

DNSResourceRecord:ttl() -> int
TTL (Time To Live) of this record

DNSResourceRecord:signttl() -> int
If non-zero, TTL that will be used in the RRSIG of the record

DNSResourceRecord:domainId() -> int
Backend related domain ID of the zone the record belongs to

DNSResourceRecord:qtype() -> int
Type of the record (A, CNAME, MX, . . .)

DNSResourceRecord:qclass() -> int
Class of the record (IN, CH, . . .)

DNSResourceRecord:scopeMask() -> int

DNSResourceRecord:auth() -> bool

DNSResourceRecord:disabled() -> bool

DNSRecord objects

A *DNSRecord* object represents a record. Creating a DNSRecord is done with the *newDR()*.

Functions and methods of a DNSRecord

newDR(*name*, *type*, *ttl*, *content*, *place*) → DNSRecord
Returns a new *DNSRecord* object.

Parameters

- **name** (DNSName) – The name to the new record
- **type** (*string*) – The type of the record
- **ttl** (*int*) – The TTL of the record
- **content** (*string*) – The content of the record
- **place** (*int*) – The place where the record is located (as an integer, see *DNSRecord.place*)

class DNSRecord
A DNSRecord object represents a DNS record.

name
The name of the record. A *DNSName*.

place
The place where the record is located, you can use the following constants

- *pdns.place.QUESTION* for the question section
- *pdns.place.ANSWER* for the answer section
- *pdns.place.AUTHORITY* for the authority section
- *pdns.place.ADDITIONAL* for the additional section

ttl
The TTL of the record

type
The type of the record (as an integer). Can for example be compared to pdns.A

DNSRecord:getContent() -> string
Return a string representation of the record content

DNSRecord:getCA() -> ComboAddress
If the record type is A or AAAA, a *ComboAddress* representing the content is returned, nil otherwise

DNSRecord:changeContent(newcontent)
Replace the record content with newcontent. The type and class cannot be changed.

Parameters **newcontent** (*str*) – The replacing content

Netmasks and NetMaskGroups

There are two classes in the PowerDNS Recursor that can be used to match IP addresses against.

Netmask class

The *Netmask* class represents an IP netmask.

```
mask = newNetmask("192.0.2.1/24")
mask:isIPv4() -- true
mask:match("192.0.2.8") -- true
```

newNetmask (*mask*) → Netmask
Creates a new *Netmask*.

Parameters **mask** (*str*) – The mask to convert.

class Netmask
Represents a netmask.

Netmask:empty() -> bool
True if the netmask doesn't contain a valid address.

Netmask:getBits() -> int
The number of bits in the address.

Netmask:getNetwork() -> ComboAddress
Returns a *ComboAddress* representing the network (no mask applied).

Netmask:getMaskedNetwork() -> ComboAddress
Returns a *ComboAddress* representing the network (truncating according to the mask).

Netmask:isIpv4() -> bool

Deprecated since version v4.3.0: True if the netmask is an IPv4 netmask.

Netmask:isIPv4() -> bool

New in version v4.3.0: True if the netmask is an IPv4 netmask.

Netmask:isIpv6() -> bool

Deprecated since version v4.3.0: True if the netmask is an IPv6 netmask.

Netmask:isIPv6() -> bool

Deprecated since version v4.3.0: True if the netmask is an IPv6 netmask.

Netmask:match(address) -> bool
True if the address passed in address matches

Parameters **address** (*str*) – IP Address to match against.

Netmask:toString() -> str
Returns a human-friendly representation.

NetMaskGroup class

NetMaskGroups are more powerful than plain Netmasks. They can be matched against netmasks objects:

```
nmg = newNMG()
nmg:addMask("127.0.0.0/8")
nmg:addMasks({"213.244.168.0/24", "130.161.0.0/16"})
nmg:addMasks(dofile("bad-ips.lua")) -- contains return {"ip1","ip2"..}

if nmg:match(dq.remoteaddr) then
  print("Intercepting query from ", dq.remoteaddr)
end
```

Prefixing a mask with `!` excludes that mask from matching.

`newNMG`([*masks*]) → NetMaskGroup

Changed in version 4.5.0: Added the optional `masks` parameter.

Returns a new *`NetMaskGroup`*. If no masks are passed, the object is empty.

Parameters **`masks`** (*{str}*) – The masks to add.

`class NetMaskGroup`

IP addresses are passed to Lua in native format.

`NetMaskGroup:addMask(mask)`

Adds `mask` to the NetMaskGroup.

Parameters **`mask`** (*str*) – The mask to add.

`NetMaskGroup:addMasks(masks)`

Adds `masks` to the NetMaskGroup.

Parameters **`masks`** (*{str}*) – The masks to add.

`NetMaskGroup:match(address) -> bool`

Returns true if `address` matches any of the masks in the group.

Parameters **`address`** (`ComboAddress`) – The IP address to match the netmasks against.

QType objects

The QType class lets you deal easily with the different kind of resource types, like 'A', 'NS', 'CNAME', etc. These types have both a name and a number. This class helps seamlessly move between them.

Functions and methods of a `QType`

`newQType`(*name*) → QType

Returns a new QType object from *name*. Name can either contain the code of the type prefixed with a sharp character, or its name directly

Parameters **`name`** (*string*) – The name of the QType

```
type = newQType("CNAME")
anothertype = newQType("#5")
if type == anothertype then -- prints "equal!"
  print('equal!')
end
```

`class QType`

It has these methods:

`QType:getCode() -> int`

Returns the numeric code corresponding to the type

`QType:getName() -> string`

Returns the name of the type

Other functions

pdnslog(*message*[, *loglevel*])

Log the *message* at the daemon level

Parameters

- **message** (*string*) – The message to log
- **loglevel** (*int*) – The urgency level of the message. Defaults to *pdns.loglevels.Warning*

You can use the following constants as log levels :

- *pdns.loglevels.Alert*
- *pdns.loglevels.Critical*
- *pdns.loglevels.Debug*
- *pdns.loglevels.Emergency*
- *pdns.loglevels.Error*
- *pdns.loglevels.Info*
- *pdns.loglevels.Notice*
- *pdns.loglevels.Warning*

pdnsrandom([*upper_bound*])

Get a random number.

Parameters **upper_bound** (*int*) – The upper bound. You will get a random number below this upper bound.

CHAPTER

FIFTEEN

GUIDES AND HOW TOS

15.1 Basic setup: configuring database connectivity

This shows you how to configure the Generic SQLite3 backend.

Make sure the SQLite3 backend is installed:

```
sudo apt-get install pdns-backend-sqlite3
```

or

```
sudo yum install pdns-backend-sqlite
```

This backend is called *'gsqlite3'*, and needs to be configured in `pdns.conf`. Add the following lines, adjusted for your local setup:

```
launch=gsqlite3
gsqlite3-database=/var/lib/powerdns/pdns.sqlite3
```

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

Now create the database (on RPM systems, the schema path is `/usr/share/doc/pdns-backend-sqlite/schema.sqlite3.sql`):

```
sudo mkdir /var/lib/powerdns
sudo sqlite3 /var/lib/powerdns/pdns.sqlite3 < /usr/share/doc/pdns-backend-sqlite3/
↪schema.sqlite3.sql
sudo chown -R pdns:pdns /var/lib/powerdns
```

And start PowerDNS

```
sudo systemctl start pdns
```

or

```
sudo systemctl restart pdns
```

Make sure no error is reported, and use `systemctl status pdns` to make sure PowerDNS was started correctly.

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

```
$ dig a www.example.com @127.0.0.1

; <<>> DiG 9.10.3-P4-Debian <<>> a www.example.com @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 40870
...
```

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

Note the `REFUSED` status - this is the code most name servers use to indicate they do not know about a domain.

Now, let's add a zone and some records:

```
$ sudo -u pdns pdnsutil create-zone example.com ns1.example.com
Creating empty zone 'example.com'
Also adding one NS record
$ sudo -u pdns pdnsutil add-record example.com '' MX '25 mail.example.com'
New rrset:
example.com. 3005 IN MX 25 mail.example.com
$ sudo -u pdns pdnsutil add-record example.com. www A 192.0.2.1
New rrset:
www.example.com. 3005 IN A 192.0.2.1
```

This should be done as the `pdns` user (or root), as sqlite3 requires write access to the directory of the database file.

Note: `pdnsutil` is a tool that can manipulate zones, set DNSSEC parameters for zones and does *many other* things. It is *highly* recommended to use `pdnsutil` or the *HTTP API* to modify zones instead of using raw SQL, as `pdnsutil` and the API perform checks on the data and post-store changes to prevent issues when serving DNS data.

If we now requery our database, `www.example.com` should be present:

```
$ dig +short www.example.com @127.0.0.1
192.0.2.1

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

If this is not the output you get, remove `+short` to see the full output so you can find out what went wrong. The first problem could be that PowerDNS has a *Packet Cache* and a *Query Cache* for performance reasons. If you see old, or no, data right after changing records, wait for *cache-ttl*, *negquery-cache-ttl*, *query-cache-ttl*, or *zone-cache-refresh-interval* to expire before testing.

Now, run `pdnsutil edit-zone example.com` and try to add a few more records, and query them with dig to make sure they work.

You now have a working database driven nameserver!

To convert other zones already present, see the *migration guide*.

15.1.1 Common problems

Most problems involve PowerDNS not being able to connect to the database. This section covers more than just SQLite.

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.

15.1.2 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

You currently don't have a backend configured in the configuration file. Add a *launch* statement for the backend you want to use.

If you are following this guide and using an sqlite database as a backend, please add the `launch=gsqlite3` instruction to 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.

15.2 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.

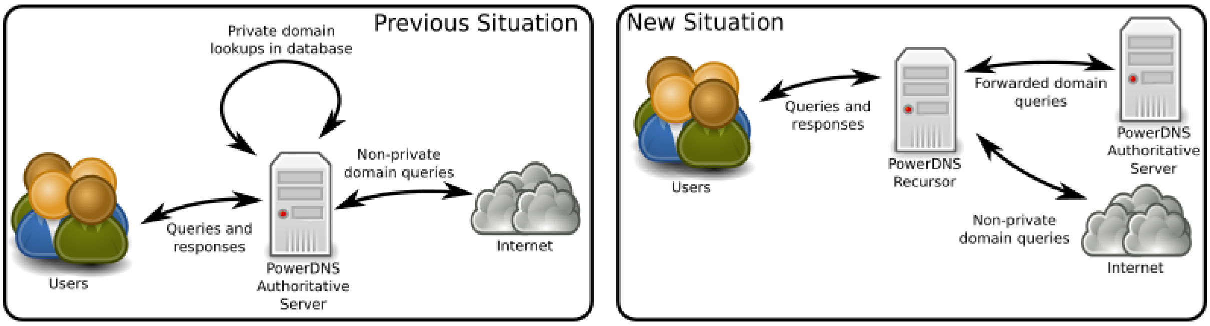
The first scenario is the one where the Authoritative Server is used as a recursor with some private domains for trusted clients. The second scenario is the one where the Authoritative Server serves publicly available domains and is a recursor for a subset of clients.

15.2.1 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.


Previous Situation
Private domain lookups in database
Non-private domain queries
Queries and responses
Users
PowerDNS Authoritative Server
Internet
New Situation
Queries and responses
Forwarded domain queries
Users
PowerDNS Recursor
PowerDNS Authoritative Server
Non-private domain queries
Internet

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-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 cannot 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..
```

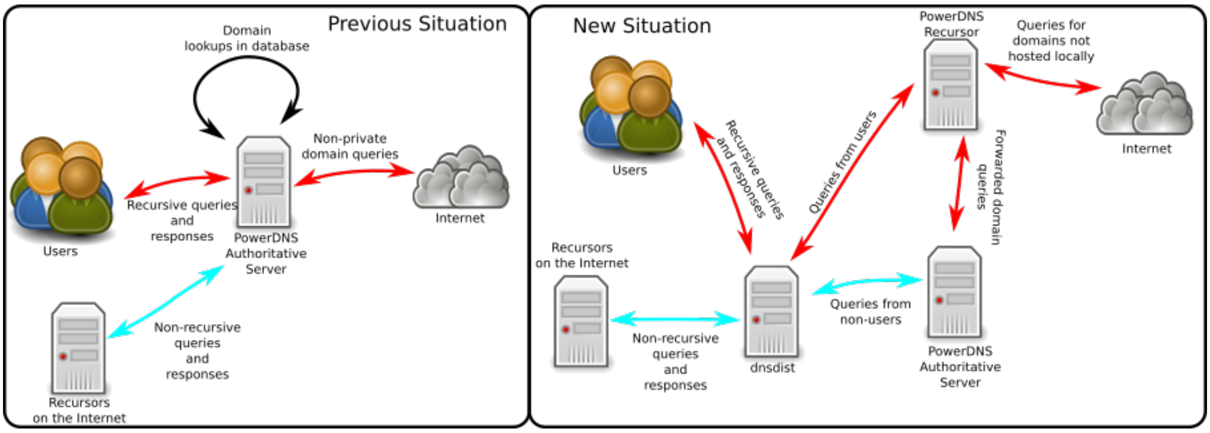
4. Restart the Authoritative Server and the Recursor

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

15.2.2 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.


Previous Situation
Domain lookups in database
Users
Recursive queries and responses
PowerDNS Authoritative Server
Non-private domain queries
Internet
Non-recursive queries and responses
Recursors on the Internet
New Situation
Users
Recursive queries and responses
Queries from users
PowerDNS Recursor
Queries for domains not hosted locally
Internet
Forwarded domain queries
Recursors on the Internet
Non-recursive queries and responses
dnsdist
Queries from non-users
PowerDNS Authoritative Server

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-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 cannot 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	dnsdist 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')
```

(continues on next page)

(continued from previous page)

```
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 dnsdist

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

15.3 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`.

15.3.1 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!

15.3.2 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
```

15.4 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. If the ALIAS target cannot be resolved (SERVFAIL) or does not exist (NXDOMAIN) the authoritative server will answer SERVFAIL.

15.4.1 AXFR Zone transfers

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 look up the A and AAAA records of the name in the ALIAS-record and send the results in the AXFR. This is useful when your secondary servers do not understand ALIAS, or should not look up the addresses themselves. Note that secondaries will not automatically follow changes in those A/AAAA records unless you AXFR regularly.

If the ALIAS target cannot be resolved, the AXFR will fail. When set to 'ignore-errors', an unresolvable ALIAS target will be omitted from the outgoing transfer.

Warning: Setting `setting-outgoing-axfr-expand-alias` to 'ignore-errors', will allow an outgoing AXFR with a broken ALIAS target to complete, but the secondary server will receive an incomplete zone. There is no standard mechanism for automatic re-transfer for zones broken in this way. You should make sure this behaviour is acceptable in your use case, provide custom integration tooling to monitor such problems, and possibly fix them automatically.

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.

15.4.2 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.

15.5 Using SVCB and derived records

The PowerDNS Authoritative Server has support for the SVCB record and derived records (like HTTPS). This support includes doing the standards recommended following of alias-form records in-zone and adding those to the additional section. Apart from that, there's the PowerDNS special for "autohints".

15.5.1 Automatic hints

PowerDNS can automatically fill in `ipv4hint` and `ipv6hint` parameters in SVCB records based on A and AAAA records already present in the zone. This can be enabled by setting *svc-autohints* to 'yes'.

New in version 4.5.0: The `svc-autohints` setting was added in 4.5.0

Consider the following zone content:

```
example.org      IN HTTPS 0 www.example.org

www.example.org  IN HTTPS 1 . ipv4hint=auto
www.example.org  IN A     192.0.2.1
www.example.org  IN AAAA  2001:db8::1

ipv6.example.org IN HTTPS 1 . ipv6hint=auto
ipv6.example.org IN AAAA  2001:db8::2
ipv6.example.org IN AAAA  2001:db8::3
```

A query for `example.org|HTTPS` will be responded to like this:

```
;; QUESTION SECTION:
;example.org.                 IN      HTTPS

;; ANSWER SECTION:
example.org.          3600    IN      HTTPS   0 www.example.org

;; ADDITIONAL SECTION:
www.example.org.      3600    IN      A       192.0.2.1
www.example.org.      3600    IN      HTTPS   1 . ipv4hint=192.0.2.1
www.example.org.      3600    IN      AAAA    2001:db8::1
```

Notice that PowerDNS did additional processing and added the target HTTPS record, and the address records to the additional section. Also notice that the ipv4hint in the `www.example.org|HTTPS` records is set to the value of the A record.

PowerDNS takes all the corresponding address records, e.g. when asking for ipv6.example.org:

```
;; QUESTION SECTION:
;ipv6.example.org.            IN      HTTPS

;; ANSWER SECTION:
ipv6.example.org.     3600    IN      HTTPS   1 . ipv6hint=2001:db8::2,2001:db8::3

;; ADDITIONAL SECTION:
ipv6.example.org.     3600    IN      AAAA    2001:db8::2
ipv6.example.org.     3600    IN      AAAA    2001:db8::3
```

When autohints are applied

PowerDNS expands the autohints both when answering a query, as well as when serving an AXFR. The text "auto" is **never** served over the wire, ensuring compatibility with any and all client software.

When no address records exist

Consider the following zone content:

```
no-ipv6.example.org  IN HTTPS 1 . ipv4hint=auto ipv6hint=auto
no-ipv6.example.org  IN A     192.0.2.2
```

Here, no AAAA record exists for www.example.org, so PowerDNS cannot put any data in the ipv6hint. In this case, the ipv6hint parameter is dropped when answering the query (and on AXFR):

```
;; QUESTION SECTION:
;no-ipv6.example.org.        IN      HTTPS

;; ANSWER SECTION:
no-ipv6.example.org.  3600    IN      HTTPS   1 . ipv4hint=192.0.2.2

;; ADDITIONAL SECTION:
no-ipv6.example.org.  3600    IN      A       192.0.2.2
```

pdnsutil checks if the autohints in SVCB and derived records can be found in the zone when using `pdnsutil check-zone`. It will emit a warning when there are no hints to be found:

```
[warning] HTTPS record for no-ipv6.example.org has automatic IPv6 hints, but no␣
↪AAAA-record for the target at no-ipv6.example.org exists.
```

When autohints exist but are disabled

When *svc-autohints* is not enabled, the parameter is dropped when its value is `auto`.

15.6 KSK Rollover

Before attempting a KSK rollover, please read **RFC 6781 "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" scheme from the above mentioned RFC, as specified in **section 4.1.2**. Phases are named after the steps in the diagram in that section.

After every change, use your favourite DNSSEC checker (DNSViz, VeriSign DNSSEC Analyzer, a validating resolver) to make sure no mistakes have crept in.

> **Warning:** For every mutation to your zone make sure that your serial is bumped, so your secondaries pick up the changes too. If you are using AXFR replication, this usually is as simple as `pdnsutil increase-serial example.com`

15.6.1 Phase: Initial

In the `initial` situation, we have a KSK and the parent zone contains a DS matching that KSK. Assuming this situation has existed for a few days, or perhaps way longer, we can move on to the `new DNSKEY` phase without delay.

15.6.2 Phase: new DNSKEY

At first note down algorithm of currently used KSK, because new KSK shall use the same one, by running following command:

```
pdnsutil show-zone example.com
```

To create a new **active** and **published** KSK with the same algorithm for the zone, run something like:

```
pdnsutil add-zone-key example.com ksk active published ALGORITHM
```

Please note down the key ID that `add-zone-key` reports. You can also retrieve it later with `pdnsutil show-zone example.com`.

After this the DNSKEY set will be signed by both KSKs.

Please check that your secondaries now show both the old and new DNSKEYs when queried for them with `dig DNSKEY example.com @...`.

Now that the new DNSKEY is active and published, we need to wait for caches to pick it up. Check the DNSKEY TTL and then wait for at least that long.

15.6.3 Phase: DS change

The DNSKEY set is currently signed with both KSKs and keys of both are published. This means that a DS for either old or new KSK is sufficient for validation. We can now switch the DS record in the parent zone - there is no need to have DSes for both KSKs in the parent zone.

Using `pdnsutil show-zone example.com` or `pdnsutil export-zone-ds example.com`, extract the DNSKEY or DS for new KSK, depending on what the parent zone operator takes as input. Note that these commands print DNSKEYs and/or DSes for both the old and the new KSK.

Check the DS TTL at the parent, for example: `dig DS example.com @c.gtld-servers.net` for a delegation from `.com`.

Submit the new DNSKEY and/or DS for of new KSK to the parent, and make sure to delete those for the old KSK.

Check again with the parent to see whether the new DS is published.

Then, wait for at least as long as the TTL for the old DS was.

15.6.4 Phase: DNSKEY removal

The parent DS is pointing at the new KSK and the old DS has expired from all caches. However, both sets of DNSKEYs are still in caches. It is time to remove the old DNSKEY:

```
pdnsutil remove-zone-key example.com OLD_KSK_ID
```

Please check that your secondaries now only show the new set of keys when queried with `dig DNSKEY example.com @...`.

15.6.5 Conclusion

After at least another DNSKEY TTL time the old DNSKEY shall expire from caches.

Your KSK Rollover is complete.

15.7 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!

15.8 ZSK Rollover

Before attempting a ZSK rollover, please read **RFC 6781 "DNSSEC Operational Practices, Version 2", section 4** carefully to understand the terminology, actions and timelines (TTL and RRSIG expiry) involved in rolling a ZSK.

This How To describes the "Pre-Publish" approach from the above mentioned RFC, as specified in **section 4.1.1.1**. Phases are named after the steps in the diagram in that section.

Warning: The following instructions assume rollover of a key which is NOT a Secure Entry Point (SEP), please confirm this fact before proceeding any further.

After every change, use your favourite DNSSEC checker (DNSViz, VeriSign DNSSEC Analyzer, a validating resolver) to make sure no mistakes have crept in.

Warning: For every mutation to your zone make sure that your serial is bumped, so your secondaries pick up the changes too. If you are using AXFR replication, this usually is as simple as `pdnsutil increase-serial example.com`

15.8.1 Phase: Initial

In the `initial` situation, we have old ZSK key used to sign all the data in the zone. Assuming this situation has existed for a few days, or perhaps way longer, we can move on to the `new DNSKEY` phase without delay.

15.8.2 Phase: new DNSKEY

At first note down algorithm of currently used ZSK, because new ZSK shall use the same one, by running following command:

```
pdnsutil show-zone example.com
```

To create a new **inactive** but **published** ZSK with the same algorithm, run something like:

```
pdnsutil add-zone-key example.com zsk inactive published ALGORITHM
```

Please note down the key ID that `add-zone-key` reports. You can also retrieve it later with `pdnsutil show-zone example.com`.

PowerDNS will now publish the new DNSKEY while the old DNSKEY remains published and active for signing.

Please check that your secondaries now show both the old and new DNSKEYs when queried for them with `dig DNSKEY example.com @....`

Now that the new DNSKEY is published, we need to wait for caches to pick it up. Check the DNSKEY TTL and then wait at least that long.

15.8.3 Phase: new RRSIGs

To change the RRSIGs on records in the zone, the new DNSKEY must be made active and the old DNSKEY must be made inactive.

```
pdnsutil activate-zone-key example.com NEW-ZSK-ID
pdnsutil deactivate-zone-key example.com OLD-ZSK-ID
```

After this, PowerDNS will sign all records in the zone with the new ZSK and remove all signatures made with the old ZSK.

Please check that your secondaries now show only the new signatures.

In your zone, check for the highest TTL you can find. This includes the SOA TTL and the SOA MINIMUM, which affect negative caching, including NSEC/NSEC3 records. *The DNSKEY TTL is also taken from the SOA MINIMUM.*

Now wait for at least that long. Depending on your setup, this will usually be between a few hours and a few days.

15.8.4 Phase: DNSKEY removal

The last step is to remove the old DNSKEY from the zone:

```
pdnsutil remove-zone-key example.com OLD-ZSK-ID
```

Please check that your secondaries now show only the new DNSKEY when queried with `dig DNSKEY example.com @....`

15.8.5 Conclusion

After at least another DNSKEY TTL time the old DNSKEY shall expire from caches.

Your ZSK Rollover is complete.

15.9 Algorithm Rollover

Before attempting an algorithm rollover, please read **RFC 6781 "DNSSEC Operational Practices, Version 2", section 4** carefully to understand the terminology, actions and timelines (TTL and RRSIG expiry) involved in changing DNSKEY algorithms.

This How To describes the "conservative" approach from the above mentioned RFC, as specified in **section 4.1.4**. Phases are named after the steps in the diagram in that section. The following instruction assume a KSK+ZSK setup; if you only have a CSK, ignore the ZSK steps.

After every change, use your favourite DNSSEC checker (DNSViz, VeriSign DNSSEC Analyzer, a validating resolver) to make sure no mistakes have crept in.

During this process, response sizes will be larger than usual, due to double sets of signatures, and double the amount of DNSKEYs. Please check that these bigger packets can make it out of your network without trouble, and verify that you and your secondaries can serve queries over TCP as well.

Warning: For every mutation to your zone (so, every step except updating DS in the parent), make sure that your serial is bumped, so your secondaries pick up the changes too. If you are using AXFR replication, this usually is as simple as `pdnsutil increase-serial example.com`

15.9.1 Phase: initial

In the `initial` situation, we have a KSK+ZSK with our old algorithm, and the parent zone contains a DS matching that KSK. Assuming this situation has existed for a few days, or perhaps way longer, we can move on to the `new RRSIGs` phase without delay.

15.9.2 Phase: new RRSIGs

To create signatures with the new algorithm, without publishing keys, run something like:

```
pdnsutil add-zone-key example.com KSK active unpublished ecdsa384
pdnsutil add-zone-key example.com ZSK active unpublished ecdsa384
```

Note the key IDs that add-zone-key reports. You can also retrieve these later with `pdnsutil show-zone example.com`.

After this, PowerDNS will sign all records in the zone with both the old and new ZSKs, and the DNSKEY set will be signed by both KSKs.

Please check that your secondaries also show the new signatures.

In the next step, we will publish the new keys. Once we do that, some validators will demand that those new keys also sign all records in the zone. However, those validators may still have records signed with only the old ZSK in cache. So, we need to wait for those records to expire.

In your zone, check for the highest TTL you can find. This includes the SOA TTL and the SOA MINIMUM, which affect negative caching, including NSEC/NSEC3 records. *The DNSKEY TTL is also taken from the SOA MINIMUM.*

Now, wait for that long. Depending on your setup, this will usually be between a few hours and a few days.

15.9.3 Phase: new DNSKEY

In the previous step, we generated two new keys, and signed our zone with them, without actually exposing validators to the new keys. After waiting for all records in our zone to expire from caches, we can publish the DNSKEYs:

```
pdnsutil publish-zone-key example.com 3
pdnsutil publish-zone-key example.com 4
```

Replace `3` and `4` with the key IDs gathered in the previous step, or find them in `pdnsutil show-zone example.com`. PowerDNS will now publish the new DNSKEYs that have already been used for signing for a while. The old DNSKEYs remain published, and active for signing, for now.

Please check that your secondaries now show both the old and new DNSKEYs when queried for them with `dig DNSKEY example.com @...`.

Now that the new DNSKEYs are published, we again need to wait for caches to pick them up, before we switch DS records in the parent.

Check the DNSKEY TTL - then wait that long.

15.9.4 Phase: new DS

Our zone is currently fully signed with two algorithms, and keys for both algorithms are published. This means that a DS for either the old or new algorithm is sufficient for validation. We can now switch the DS - there is no need to have DSes for both algorithms in the parent zone.

Using `pdnsutil show-zone example.com` or `pdnsutil export-zone-ds example.com`, extract the new DNSKEYs or new DSes, depending on what the parent zone operator takes as input. Note that these commands print DNSKEYs and/or DSes for both the old and the new algorithm.

Check the DS TTL at the parent, for example: `dig DS example.com @c.gtld-servers.net` for a delegation from `.com`.

Submit the new algorithm DNSKEY/DSes to the parent, and make sure to delete those for the old algorithm.

Check again with the parent to see whether the new DS is published.

Then, wait for as long as the TTL on the old DS was.

15.9.5 Phase: DNSKEY removal

We are signing with two algorithms. The parent DS is pointing at the KSK for the new algorithm, and the old DS has expired from all caches. However, both sets of DNSKEYs are still in caches. It is time to remove the old DNSKEYs, while keeping their signature:

```
pdnsutil unpublish-zone-key example.com 1
pdnsutil unpublish-zone-key example.com 2
```

Replace `1` and `2` with the IDs of the old keys.

Please check that your secondaries now only show the new set of keys when queried with `dig DNSKEY example.com @...`.

Over the next DNSKEY TTL seconds, validators can still have both sets of keys in cache. So, we leave our signatures in until that time passes.

15.9.6 Phase: RRSIGs removal

After waiting DNSKEY TTL seconds, caches should only have a copy of our new set of keys. This means we can now safely stop signing with the old keys:

```
pdnsutil deactivate-zone-key example.com 1
pdnsutil deactivate-zone-key example.com 2
```

Alternatively, you can use `remove-zone-key` to remove all traces of the old keys.

15.9.7 Conclusion

In another hours-to-a-few-days, the old signatures will expire from caches.

Your algorithm roll is complete.

15.10 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().

And that's it. For completeness, add TLSA and 52 to the QType enum in qtype.hh, which makes it easier to refer to the TLSA record in code if so required.

CHAPTER

SIXTEEN

BACKENDS

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

Name	Native	Primary	Secondary	Producer	Consumer	Autoprimary	DNS Update	DNSSEC	Launch
BIND	Yes	Yes	Yes	No	No	Yes	No	Yes	bind
Generic Mysql/Mariadb	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	gmysql
Generic ODBC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	godbc
Generic Postgresql	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	gpgsql
Generic SQLite3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	gsqlite3
GeoIP	Yes	No	No	No	No	No	No	Yes	geoip
LDAP	Yes	Yes	No	No	No	No	No	No	ldap
LMDB	Yes	Yes	Yes	Yes	Yes	No	No	Yes	lmdb
Lua2	Yes	Yes	No	No	No	No	No	Yes	lua2
Pipe	Yes	No	No	No	No	No	No	No	pipe
Random	Yes	No	No	No	No	No	No	No	random
Remote	Yes	Yes*	Yes*	No	No	Yes*	No	Yes*	remote
TinyDNS	Yes	Yes	No	No	No	No	No	No	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.

16.1 BIND zone file backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Experimental
- DNSSEC: Yes
- Disabled data: No
- Comments: No
- API: Read-only
- Zone caching: Yes (except in hybrid mode)
- Module name: bind
- Launch: `bind`

The BIND backend 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 BIND backend 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.

Unique to this PowerDNS backend is that it serves from plain zone files, which allows for hand-crafting zone files, only takes a tiny footprint in terms of server resource usage while being *performant efficiently*.

Note: Because this backend retrieves its configuration from plain files and not a database, the HTTP API is unable to process changes for this backend. This effectively makes the API read-only for zones hosted by the BIND backend.

16.1.1 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`
    - `primaries` (added in version 4.9.0)
    - `also-notify`

Unknown directives will be ignored.

`bind-check-interval`

Interval in seconds to check for zone file changes. Default is 0 (disabled).

See *Operation* section for more information.

`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.

You can use `pdnsutil create-bind-db` to make this database file for you.

Warning: If this is left empty on slaves and a presigned zone is transferred, it will (silently) serve it without DNSSEC. This in turn results in serving the domain as bogus.

`bind-dnssec-db-journal-mode`

SQLite3 journal mode to set. The default is WAL. Set to empty to leave the journal mode alone.

`bind-hybrid`

Store DNSSEC keys and metadata storage in another 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.

Autoprimary support (experimental)

`bind-autoprimaries`

Changed in version 4.9.0: This was called `bind-supermasters` before 4.9.0.

Specifies file where to read list of autoprimaries. BIND backend only checks IP address of primary server.

The file must contain one IP and account per line, separated by whitespace.

BIND backend can only read this file, not write it.

`bind-autoprimary-config`

Changed in version 4.9.0: This was called `bind-supermaster-config` before 4.9.0.

When a new zone is configured via the autosecondary mechanism, bindbackend *writes* a zone entry to this file.

Your `bind-config` file should have an `include` statement to make sure this file is read on startup.

`bind-autoprimary-destdir`

Changed in version 4.9.0: This was called `bind-supermaster-destdir` before 4.9.0.

Each new zone configured via the autosecondary mechanism gets a zone file in this directory. This directory must be writable.

16.1.2 Operation

On launch, the BIND backend 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 (or zone transfer) for a zone comes in, and then only after *bind-check-interval* seconds have passed since 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` command is issued.

Please note that also the *xfr-cycle-interval* setting controls how often a master would notify a slave about changes. Especially in ‘hidden master’ configurations, where servers usually don’t receive regular queries, you may want to lower that setting to a value as low as *bind-check-interval*.

16.1.3 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-extended-status [domain ...]`

New in version 4.3.0.

Output an extended status of a domain or domains, containing much more information than the simple domain status, like the number of records currently loaded, whether pdns is master or slave for the domain, the list of masters, various timers, etc

`bind-domain-status [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 immediately, 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.

16.1.4 Performance

The BIND backend 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`.

16.1.5 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.

16.2 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: It is highly recommended to not use raw SQL commands to manipulate records in the database. Either use the *REST-API* or (as shown in the examples here) *pdnsutil*.

If raw SQL is used, ensure that record and zone names are **never** terminated with a trailing '.'. The only exception is a root-zone, this must have the name of '.' (no quotes).

16.2.1 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:

```
pdnsutil create-zone example.com
```

Records can now be added using `pdnsutil add-record` or `pdnsutil edit-zone`.

Secondary operation

These backends are fully secondary capable. To become a secondary of the 'example.com' domain, using 198.51.100.6 as the primary execute this:

```
pdnsutil create-secondary-zone example.com 198.51.100.6
```

And wait a while for PowerDNS to pick up the addition - which happens within one minute (this is determined by the *xfr-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 secondary domains are fresh
Apr 09 13:35:29 1 secondary domain needs checking
Apr 09 13:35:29 Domain example.com is stale, primary 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 *xfr-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 primaries per zone, and also port numbers for these primaries:

```
pdnsutil create-secondary-zone example.com 198.51.100.6 2001:0DB8:15:4AF::4
pdnsutil create-secondary-zone example.net 198.51.100.20:5301
↪'[2001:0DB8:11:6E::4]:54'
```

Autoprimary operation

To configure a *autoprimary* with IP address 203.0.113.53 which lists this installation as 'autosecondary.example.com', issue the following:

```
pdnsutil add-autoprimary 203.0.113.53 autosecondary.example.com internal
```

From now on, valid notifies from 203.0.113.53 for which the zone lists an NS record containing 'autosecondary.example.com' will lead to the provisioning of a secondary domain under the account 'internal'. See *Autoprimary: automatic provisioning of secondaries* for details.

Primary operation

The generic SQL backend is fully primary 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 primary operation instead of the default native replication, issue:

```
pdnsutil create-zone example.com
pdnsutil set-kind example.com MASTER
```

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.

This value cannot be set with `pdnsutil`.

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, xfr still works. A secondary considers a disabled domain to have a serial of 0; this implies that a secondary domain will not stay disabled.

16.2.2 Handling DNSSEC signed zones

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

Rules for filling out DNSSEC fields

Note: This section is only relevant for users who use SQL to change records and zones.

Two additional fields in the 'records' table are important: 'auth' and 'ordername'. 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 'ordername' field needs to be filled out depending on the NSEC/NSEC3 mode. When running in NSEC3 'Narrow' mode, the ordername field is ignored and best left empty. In NSEC/NSEC3 mode, the ordername field should be NULL for any glue but filled in for all delegation NS records and all authoritative records. In NSEC3 opt-out mode, ordername 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 ordername.

In 'NSEC3' non-narrow mode, the ordername 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.

16.2.3 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 --launch=BACKEND --config=default`.

This section is useful for users who have their own database schema and wish to have PowerDNS use the existing database.

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 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.
- `publish-domain-key-query`: Called to set a cryptokey to published.
- `unpublish-domain-key-query`: Called to set a cryptokey to unpublished.
- `clear-domain-all-keys-query`: Called to remove all DNSSEC keys for a zone.
- `remove-domain-key-query`: Called to remove a crypto key.

Primary/secondary queries

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

On primaries

- `info-all-primary-query`: Called to get data on all domains for which the server is primary.
- `update-serial-query` Called to update the last notified serial of a primary domain.

On secondaries

- `info-all-secondaries-query`: Called to retrieve all secondary domains.
- `update-lastcheck-query`: Called to update the last time a secondary domain was successfully checked for freshness.
- `update-primary-query`: Called to update the primary address of a domain.

On autoprimary

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

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=?
```

16.3 Generic MySQL/MariaDB backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Case: All lower
- DNSSEC: Yes (set `gmysql-dnssec`)
- Disabled data: Yes
- Comments: Yes
- Zone caching: 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.

> **Warning:** While it is possible to run the Generic MySQL/MariaDB backend on top of MySQL/MariaDB views, we have received several reports of this causing performance problems and memory leaks. Please know that when reporting problems when running PowerDNS on top of a modified schema, our open source support offering requires you to reproduce your problem on an unmodified schema without views.

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 affects 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;
```

16.3.1 Using MySQL/MariaDB replication

To support `NATIVE` domains, the `binlog_format` for the MySQL/MariaDB replication **must** be set to `MIXED` or `ROW` to prevent differences in data between replicated servers. See "Setting The Binary Log Format" and "Binary Log Formats" for more information.

Otherwise, you will probably see:

```
Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT =␣
↪STATEMENT and at least one table uses a storage engine limited to row-based␣
↪logging.
InnoDB is limited to row-logging when transaction isolation level is READ␣
↪COMMITTED or READ UNCOMMITTED.
```

16.3.2 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-ssl

Deprecated since version 5.0.0.

Before 5.0.0: Send the CLIENT_SSL capability flag to the server. SSL support is announced by the server via CLIENT_SSL and is enabled if the client returns the same capability. Default: no.

5.0.0 and up: this option does nothing. Use `gmysql-group` and put your TLS settings in `my.cnf`.

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

gmysql-thread-cleanup

Older versions (such as those shipped on RHEL 7) of the MySQL/MariaDB client libraries leak memory unless applications explicitly report the end of each thread to the library. Enabling `gmysql-thread-cleanup` tells PowerDNS to call `mysql_thread_end()` whenever a thread ends.

Only enable this if you are certain you need to. For more discussion, see https://github.com/PowerDNS/pdns/issues/6231.

16.3.3 Default Schema

This is the 4.7 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(8) NOT NULL,
  notified_serial       INT UNSIGNED DEFAULT NULL,
  account               VARCHAR(40) CHARACTER SET 'utf8' DEFAULT NULL,
  options               VARCHAR(64000) DEFAULT NULL,
  catalog               VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (id)
) Engine=InnoDB CHARACTER SET 'latin1';

CREATE UNIQUE INDEX name_index ON domains(name);
CREATE INDEX catalog_idx ON domains(catalog);


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,
  disabled              TINYINT(1) DEFAULT 0,
  ordername             VARCHAR(255) BINARY DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```
  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,
  published             BOOL DEFAULT 1,
  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),
```

(continues on next page)

(continued from previous page)

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

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

16.4 Generic ODBC Backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- Case: All lower
- DNSSEC: Yes
- Disabled data: Yes
- Comments: Yes
- Zone caching: 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 gmysql or gsqlite3), it is highly recommended to use that backend instead!

16.4.1 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.

16.4.2 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 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.

16.4.3 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. FreeDTS 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`).

This is the schema for 4.3. The 4.2 schema and the 4.1 schema is available on GitHub.

```
CREATE TABLE domains (
  id                    INT IDENTITY,
  name                  VARCHAR(255) NOT NULL,
  master                VARCHAR(128) DEFAULT NULL,
  last_check            INT DEFAULT NULL,
  type                  VARCHAR(8) NOT NULL,
  notified_serial       INT DEFAULT NULL,
  account               VARCHAR(40) DEFAULT NULL,
  options               VARCHAR(MAX) DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```
  catalog               VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (id)
);

CREATE UNIQUE INDEX name_index ON domains(name);
CREATE INDEX catalog_idx ON domains(catalog);


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,
  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),
  PRIMARY KEY (id)
);

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

(continues on next page)

(continued from previous page)

```
CREATE TABLE cryptokeys (
  id                      INT IDENTITY,
  domain_id               INT NOT NULL,
  flags                   INT NOT NULL,
  active                  BIT,
  published               BIT DEFAULT 1,
  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.

16.5 Generic PostgreSQL backend

- Native: Yes
- Master: Yes
- Slave: Yes

- Superslave: Yes
- Case: All lower
- DNSSEC: Yes (set `gpgsql-dnssec`)
- Disabled data: Yes
- Comments: Yes
- Zone caching: Yes
- Module name: gpgsql
- Launch name: `gpgsql`

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 `--gpgsql` 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.

16.5.1 Settings

`gpgsql-host`

Host (ip address) to connect to. If `pgsql-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.

`gpgsql-port`

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

`gpgsql-dbname`

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

`gpgsql-user`

User to connect as. Default: not set.

`gpgsql-password`

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

`gpgsql-dnssec`

Enable DNSSEC processing for this backend. Default: no.

gpgsql-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: "".

gpgsql-prepared-statements

Prepare statements for better performance, instead of sending parameterized queries. Might not work with connection poolers. Default: yes.

New in version 4.4.0.

16.5.2 Default schema

This is the 4.7 schema.

```
CREATE TABLE domains (
  id                    SERIAL PRIMARY KEY,
  name                  VARCHAR(255) NOT NULL,
  master                VARCHAR(128) DEFAULT NULL,
  last_check            INT DEFAULT NULL,
  type                  TEXT NOT NULL,
  notified_serial       BIGINT DEFAULT NULL,
  account               VARCHAR(40) DEFAULT NULL,
  options               TEXT DEFAULT NULL,
  catalog               TEXT DEFAULT NULL,
  CONSTRAINT c_lowercase_name CHECK (((name)::TEXT = LOWER((name)::TEXT)))
);

CREATE UNIQUE INDEX name_index ON domains(name);
CREATE INDEX catalog_idx ON domains(catalog);


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,
  prio                  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,
```

(continues on next page)

(continued from previous page)

```
  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,
  published             BOOL DEFAULT TRUE,
  content               TEXT
);

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);
```

16.5.3 CockroachDB

CockroachDB is a highly available, resilient database that focuses on scaling and consistency. Specifically: it offers a PostgreSQL like database interface, which means that most tools that talk the PostgreSQL protocol can use it.

A few changes are needed on top of the generic PostgreSQL settings. CockroachDB does not natively support the range operators that some PowerDNS database queries use, and care must be taken that table index columns do not exceed the internal maximum integer size that PowerDNS uses.

Schema differences

Given the normal pgsql schema, change the following:

1. Add explicit SEQUENCEs for all SERIAL columns:

```
CREATE SEQUENCE domain_id MAXVALUE 2147483648;
CREATE SEQUENCE record_id MAXVALUE 2147483648;
CREATE SEQUENCE comment_id MAXVALUE 2147483648;
CREATE SEQUENCE meta_id MAXVALUE 2147483648;
CREATE SEQUENCE key_id MAXVALUE 2147483648;
CREATE SEQUENCE tsig_id MAXVALUE 2147483648;
```

2. Change all SERIAL / BIGSERIAL columns to use the SEQUENCEs. For instance, change

```
CREATE TABLE domains (
  id SERIAL PRIMARY KEY,
)
```

to

```
CREATE TABLE domains (
  id INT DEFAULT nextval('domain_id') PRIMARY KEY,
);
```

3. Do **not** add the following index to the records table, the text_pattern_ops operator class is not supported:

```
CREATE INDEX recordorder ON records (domain_id, ordername text_pattern_ops);
```

Configuration changes

Four queries must be overridden in the PowerDNS config, because by default they use a range operator that is not supported. These modified queries are actually taken from the generic MySQL backend, and modified for syntax:

```
gpgsql-get-order-first-query=select ordername from records where domain_id = $1␣
↪and disabled = false and ordername is not null order by 1 asc limit 1
gpgsql-get-order-before-query=select ordername, name from records where ordername
↪<= $1 and domain_id = $2 and disabled = false and ordername is not null order by␣
↪1 desc limit 1
gpgsql-get-order-after-query=select ordername from records where ordername > $1␣
↪and domain_id = $2 and disabled = false and ordername is not null order by 1 asc␣
↪limit 1
gpgsql-get-order-last-query=select ordername, name from records where ordername !=
↪'' and domain_id = $1 and disabled = false and ordername is not null order by 1␣
↪desc limit 1
```

References

See this GitHub issue for the original tests and a full working schema.

16.6 Generic SQLite 3 backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: Yes
- DNSSEC: Yes
- Disabled data: Yes
- Comments: Yes
- Zone caching: 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.

16.6.1 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 in 4.7.

```
PRAGMA foreign_keys = 1;

CREATE TABLE domains (
  id                      INTEGER PRIMARY KEY,
  name                    VARCHAR(255) NOT NULL COLLATE NOCASE,
  master                  VARCHAR(128) DEFAULT NULL,
  last_check              INTEGER DEFAULT NULL,
  type                    VARCHAR(8) NOT NULL,
  notified_serial         INTEGER DEFAULT NULL,
  account                 VARCHAR(40) DEFAULT NULL,
  options                 VARCHAR(65535) DEFAULT NULL,
  catalog                 VARCHAR(255) DEFAULT NULL
);

CREATE UNIQUE INDEX name_index ON domains(name);
CREATE INDEX catalog_idx ON domains(catalog);


CREATE TABLE records (
  id                      INTEGER PRIMARY KEY,
  domain_id               INTEGER DEFAULT NULL,
  name                    VARCHAR(255) DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```
  type                  VARCHAR(10) DEFAULT NULL,
  content               VARCHAR(65535) DEFAULT NULL,
  ttl                   INTEGER DEFAULT NULL,
  prio                  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 records_lookup_idx ON records(name, type);
CREATE INDEX records_lookup_id_idx ON records(domain_id, name, type);
CREATE INDEX records_order_idx ON records(domain_id, 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_idx ON comments(domain_id, 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
);

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,
 published              BOOL DEFAULT 1,
 content                TEXT,
 FOREIGN KEY(domain_id) REFERENCES domains(id) ON DELETE CASCADE ON UPDATE CASCADE
);

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

(continues on next page)

(continued from previous page)

```


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.

16.6.2 Configuration Parameters

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

`gsqlite3-database`

Path to the SQLite3 database.

`gsqlite3-pragma-journal-mode`

SQLite3 journal mode to set. The default is WAL. Set to empty to leave the journal mode alone.

`gsqlite3-pragma-synchronous`

Set this to 0 for blazing speed.

`gsqlite3-pragma-foreign-keys`

Enable foreign key constraints.

`gsqlite3-dnssec`

Enable DNSSEC processing.

16.6.3 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.

16.6.4 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.

16.7 GeoIP backend

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

This backend (which is a.k.a. the YAML 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 GeoIP 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*.

16.7.1 Prerequisites

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

You must have a geoip database available. As of this 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 a 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.

16.7.2 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 suit you.

`geoip-database-files`

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 MaxMind's getting started guide.

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

Drivers and options

dat legacy libGeoIP database. Options:

- **mode** The caching mode for data, one of `standard`, `memory`, `index`, or `mmap`.

mmdb driver for libmaxminddb databases. Options:

- **mode** The caching mode for data, only `mmap` is supported
- **language** The language to use, `en` by default

`geoip-zones-file`

Specifies the full path of the zone configuration file to use. The file is re-opened with a `pdns_control reload`.

`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.

16.7.3 Zonefile format

Zone configuration files use YAML syntax. Here is simple example. Note that the `-` before certain keys is part of the syntax.

```yaml
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
```

(continues on next page)

(continued from previous page)

```
# 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'
mapping_lookup_formats: ['%cc-%re', '%cc']
custom_mapping:
  fr: eu-central
  be: eu-central
  es: eu-south
  pt: eu-south
  us-tx: us-south
  us-ca: us-south
```

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, if no TTL is specified in specific record.

records Records for this domain. Each subkey must be a fully qualified name, under which an array of records follows. Every record is then keyed by its type (e.g. `a`, `txt`) and a type may exist more than once. The content for this record may then be configured as the value. However, a record can alternatively have one or more subkeys:

content The content of the record.

ttl The TTL for this record.

weight The weight for this specific content

services Defines one or more services for querying. Each service name may have one or more placeholders.

mapping_lookup_formats Defines which format to interpolate when using the `%mp` placeholder. Each entry is looked up in the given order and stops at first match. This allows using a fine granularity, (e.g. per country), while limiting the number of records to create. You can use any placeholder, except `%mp` to avoid recursion, within the given format (e.g. %cc).

custom_mapping Defines the mapping between the lookup format and a custom value to replace `%mp` placeholder.

zones_dir Directory to load zones from. Each file must contain exactly one `zone:` object, formatted like individual domains in the example configuration above.

mapping_lookup_formats Same as per domain, but used as default value if not defined at the domain level.

custom_mapping Same as per domain, but used as default value if not defined at the domain level.

Note: For each **domain**, one record of the domain name **MUST** exist with a `soa` record.

Placeholders

Services, domains and record content can contain any number of placeholders that are replaced based on the information in the database and the query.

Following placeholders are supported, and support subnet caching with EDNS:

%% literal `%`

%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

These placeholders disable caching for the record completely:

%yy Year

%mos Month name

%mo Month

%wds Weekday name

%wd Weekday

%dd Year day

%hh Hour

%ip Client IP address

%ip4 Client IPv4 address

%ip6 Client IPv6 address

Following placeholder allows custom mapping:

%mp Use formats in `mapping_lookup_formats` and use user defined `custom_mapping`

New in version 4.4.0: These placeholders have been added in version 4.4.0:

- %mp to expand user defined custom formats.

Using the `weight` attribute

You can use record attributes to define positive and non-zero weight. If this is given, only one record per type is chosen randomly based on the weight.

Probability is calculated by summing up the weights and dividing each weight with the sum.

Responses to queries

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.

You can mix service and static records to produce the sum of these records, including apex record. For instance, this configuration will send the correct response for both A and SOA queries:

```
domains:
- domain: example.com
- ttl: 300
- records:
  geo.example.com:
    - soa: ns1.example.com hostmaster.example.com 2014090125 7200 3600 1209600 3600
    - ns: ns1.example.com
    - a: 192.0.2.1
  swe.eu.example.com:
    - a: 192.0.2.2
- services:
  geo.example.com: ['%co.%cn.example.com']
```

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.

Caching and the GeoIP Backend

The *Packet Cache* and *Query Cache* will cache the records with EDNS Client Subnet information, when provided in the response. Use of certain placeholders (described above) can disable record caching for certain resource records.

That means, if you have a record like this:

```
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 a dedicated name for it.

16.8 LDAP backend

- Native: Yes
- Master: Yes
- Slave: No
- Superslave: No
- DNSSEC: No

- Disabled data: No
- Comments: No
- Zone caching: No
- Module name: ldap
- Launch name: `ldap`

16.8.1 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
        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'
```

(continues on next page)

(continued from previous page)

```
        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 )

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'
```

(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 ( 1.3.6.1.4.1.2428.20.1.33 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'
        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
```

(continues on next page)

(continued from previous page)

```
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.46 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'
        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
```

(continues on next page)

(continued from previous page)

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.64 NAME 'SVCBRecord'
        DESC 'Service binding, draft-ietf-dnsop-svcb-https-01'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.2428.20.1.65 NAME 'HTTPSRecord'
        DESC 'HTTPS service binding, draft-ietf-dnsop-svcb-https-01'
        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 'uRIRecord'
        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 'cAARecord'
        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
        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 )
```

(continues on next page)

(continued from previous page)

```
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 $
                SVCBRecord $ HTTPSRecord $
                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.

16.8.2 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.

16.8.3 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 an `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-bindmethod`

(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" *ldap-bindmethod*.

`ldap-secret`

(default "") : Password for authentication against the object specified by ldap-binddn. Only used when *ldap-bindmethod* is "simple".

`ldap-krb5-keytab`

(default: "") : Full path to the keytab file to use to authenticate. This is only used when *ldap-bindmethod* 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 entries 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".

16.8.4 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.

16.8.5 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 DNs 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
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
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 the LDAP server.

CAUTION: `ldap-method=strict` cannot 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.

16.8.6 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.

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
```

16.8.7 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:

```
index associatedDomain pres,eq,sub
```

Furthermore, if `ldap-method=strict` is set, it uses the aRecord and aAAARecord 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:

```
index aAAARecord pres,eq
index aRecord pres,eq
```

All other attributes than associatedDomain, aRecord or aAAARecord 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 requires, 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.

16.8.8 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 aAAARecord entries must follow two restrictions: They have to be fully expanded ("FFFF::1" is not allowed and it must be "FFFF:0:0:0:0:0:0:1" 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 wildcards.

16.8.9 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.

16.9 LMDB backend

- Native: Yes
- Master: Yes
- Slave: Yes
- Superslave: No
- Case: All lower
- DNSSEC: Yes
- Disabled data: Yes
- Comments: No
- Multiple instances: No
- Zone caching: Yes
- Module name: lmdb

- Launch name: `lmdb`

> **Warning:** The LMDB backend is considered stable as of 4.4.0. Version 4.3.0 was stable but had an important known bug, that affects anybody with big records such as long TXT content.

16.9.1 Enabling the backend

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

16.9.2 Settings

`lmdb-filename`

Path to the LMDB file (e.g. */var/lib/powerdns/pdns.lmdb*)

> **Warning:** On systemd systems, 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 to the LMDB database.

`lmdb-shards`

Records database will be split into this number of shards e.g. lmdb-shards=64. Default is 2 on 32 bits systems, and 64 on 64 bits systems.

`lmdb-sync-mode`

Changed in version 4.9.0.

`mapasync` choice removed

Synchronisation mode: one of sync, nosync, nometasync (default: sync).

`sync` (default since 4.9.0) LMDB synchronous mode. Safest option, but also slightly slower. Can also be enabled with `lmdb-sync-mode=`

`nosync` don't flush systems buffers to disk when committing a transaction. This means a system crash can corrupt the database or lose the last transactions if buffers are not yet flushed to disk.

`nometasync` flush system buffers to disk only once per transaction, omit the metadata flush. This maintains database integrity, but can potentially lose the last committed transaction if the operating system crashes.

`mapasync` (default before 4.9.0) Due to a bug before version 4.9.0, this actually gave `sync` behaviour. The `mapasync` choice has been removed in version 4.9.0.

`lmdb-schema-version`

Determines the maximum schema version LMDB is allowed to upgrade to. If the on disk LMDB database has a lower version than the current version of the LMDB schema the backend will not start, unless this setting allows it to upgrade the schema. If the version of the DB is already the same as the current schema version this setting is not checked and the backend will start normally.

The default value for this setting is the highest supported schema version for the version of PowerDNS you are starting. If you want to prevent automatic schema upgrades, explicitly set this setting to the current default before upgrading PowerDNS.

PowerDNS Version	LMDB Schema version
4.2.x	1
4.3.x	2
4.4.x to 4.6.x	3
4.7.x and up	4
4.8.x and up	5

`lmdb-random-ids`

New in version 4.7.0.

- Boolean
- Default: no

Numeric IDs inside the database are generated randomly instead of sequentially. If some external process is synchronising databases between systems, this will avoid conflicts when objects (domains, keys, etc.) get added. This will also improve the detection of recreated zones for *Catalog Zones* producers.

`lmdb-map-size`

New in version 4.7.0.

Size, in megabytes, of each LMDB database. This number can be increased later, but never decreased. Defaults to 100 on 32 bit systems, and 16000 on 64 bit systems.

`lmdb-flag-deleted`

New in version 4.8.0.

- Boolean
- Default: no

Instead of deleting items from the database, flag them as deleted in the item's Lightning Stream header. Only enable this if you are using Lightning Stream.

`lmdb-lightning-stream`

New in version 4.8.0.

- Boolean
- Default: no

Run in Lightning Stream compatible mode. This:

- forces `flag-deleted` on
- forces `random-ids` on
- handles duplicate entries in databases that can result from domains being added on two Lightning Stream nodes at the same time
- aborts startup if `shards` is not set to `1`

16.9.3 LMDB Structure

PowerDNS will create the database structure, no need to manually create the database schema. Also, it is not possible to directly query the LMDB DB, so recommendation is to use either the API, or pdnsutil.

16.10 Lua2 Backend

- Native: Yes
- Master: Yes
- Slave: No
- Superslave: No
- DNSSEC: Yes
- Disabled data: No
- Comments: No
- Zone caching: Yes
- Module name: lua2
- Launch name: `lua2`

This is a rewrite of existing Lua backend. This backend is stub between your Lua script and PowerDNS authoritative server. The backend uses AuthLua4 base class, and you can use same functions and types as in any other Lua script.

Warning: Some of the function calls and configuration settings have been changed from original `Luabackend`, please review this document carefully.

Warning: All places which use DNS names now use DNSName class which cannot be compared directly to a string. To compare them against a string use either `tostring(dnsname)` or `newDN(string)`.

Warning: There is no API version 1. Use Luabackend if you need version 1.

16.10.1 API description (v2)

`bool dns_dnssec`

If your script supports DNSSEC, set this to true.

`dns_lookup(qtype, qname, domain_id, ctx)`

Perform lookup of given resource record name and type.

INPUT:

- QType qtype - Type of queried resource record
- DNSName qname - Name of queried resource record
- int domain_id - ID of associated domain
- table ctx - Query context table, contains `source_address` and `real_source_address`.

OUTPUT: Expects a array which has tables with following keys:

- DNSName name - resource record name (can also be string)
- string type - type of resource record (can also be QType or valid integer)

- string content - resource record content
- int ttl - time to live for this resource record (default: configured value)
- int domain_id - ID of associated domain (default: -1)
- bool auth - Whether data is authoritative or not (default: true)
- int last_modified - UNIX timestamp of last modification
- int scope_mask - How many bytes of source IP netmask was used for this result

NOTES: Defaults are used for omitted keys. Return empty array if you have no results. The requested record type is unlikely to match what was asked from PowerDNS. This function is **required**.

`dns_list(target, domain_id)`

List all resource records for target.

INPUT:

- DNSName target - Zone name to list
- int domain_id - Associated domain ID

OUTPUT: Same as `lookup` function. Return false if not found or wanted.

NOTES: This function is **optional**.

`dns_get_domaininfo(domain)`

Get domain information for given domain.

INPUT:

- DNSName domain - Domain to get info for

OUTPUT: Return false if not supported or found, otherwise expects a table with keys:

- string account - Associated account of this domain (default: <empty>)
- string kind - Domain kind (NATIVE,MASTER,SLAVE) (default: NATIVE)
- int id - Associated domain ID (default: -1)
- int last_check - UNIX timestamp of last check from master (default: 0)
- table of strings masters - Master servers for this domain (default: <empty>)
- long notified_serial - Notified serial to slaves (default: 0)
- long serial - Current domain serial

NOTES: This function is **optional**. Defaults are used for omitted keys. `last_check` is for automatic serial. `masters`, `account`, `notified_serial` are for master/slave interaction only. If this function is missing, it will revert into looking up SOA record for the given domain, and uses that, if found.

`dns_get_all_domains()`

Get domain information for all domains.

OUTPUT: Return false if not supported or found, otherwise return a table of string, domaininfo. See *dns_get_domaininfo()*.

NOTES: This function is **optional**, except if you need master functionality.

`dns_get_domain_metadata(domain, kind)`

Get metadata value(s) for given domain and metadata kind.

INPUT:

- DNSName domain - Domain to get metadata for
- string kind - What kind of metadata to return

OUTPUT:

- array of strings. Or false if not supported or found.

NOTES: This function is **required** if `dns_dnssec` is true.

`dns_get_all_domain_metadata(domain)`

Get all metadata for domain.

INPUT:

- DNSName domain - Domain to get metadata for

OUTPUT: Table with metadata keys containing array of strings. Or false if not supported or found.

NOTES: This function is **optional**.

`dns_get_domain_keys(domain)`

Get DNSSEC key(s) for the given domain. Content must be valid key record in format that PowerDNS understands.

INPUT:

- DNSName domain - Domain to get key(s) for

OUTPUT: Return false if not found or supported, otherwise expects array of tables with keys:

- int id - Key ID
- int flags - Key flags
- bool active - Is key active
- bool published - Is key published
- string content - Key itself

NOTES: This function is **optional**. However, not implementing this means you cannot do live signing.

`dns_get_before_and_after_names_absolute(id, qname)`

Calculate NSEC before/after value for the given qname for domain with id.

INPUT:

- int id - Associated domain id
- DNSName qname - DNS name to calculate

OUTPUT: Table with keys:

- unhashed - DNSName of the unhashed relative to domain
- before - (hashed) name of previous record relative to domain
- after - (hashed) name of next record relative to domain

NOTES: Strings are promoted to DNSNames (you can also return DNSNames directly) This function is **required** if `dns_dnssec` is true. Hashing is required with NSEC3/5. `before` and `after` should wrap, so that after record of last record is apex record. You can use `DNSName#canonCompare` to sort records in correct order.

`dns_set_notified(id, serial)`

Called after NOTIFY so that backend can store the notified serial.

INPUT:

- int id - Associated domain id
- long serial - Notified serial

NOTES: This function is **optional**. However, not implementing this can cause problems with master functionality.

16.11 Pipe Backend

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- Case: Depends
- DNSSEC: Partial, no delegation, no key storage
- Disabled data: No
- Comments: No
- Zone caching: 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 number of distributor (backend) threads (`distributor-threads`) to start per receiver thread is low by default. This can impact the performance if you have latency-bound application as backend. You should increase the number of `distributor-threads` in such cases. See *Performance and Tuning*.

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.

16.11.1 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$`. **Note**: to match the root domain, use a dot, e.g. `^\.$`

16.11.2 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   wd1.example.org IN  A   -1  203.0.113.210
DATA    ws1.example.org IN  A   3600    1   192.0.2.4
DATA    ws1.example.org IN  A   3600    1   192.0.2.5
DATA    ws1.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
```

16.11.3 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;
        }

        # 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";
```

(continues on next page)

(continued from previous page)

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


        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);

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";
```

(continues on next page)

(continued from previous page)

```
                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";
        }


        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;
```

(continues on next page)

(continued from previous page)

```
$|=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";
                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";
```

(continues on next page)

(continued from previous page)

```
                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";
}
```

16.12 Random Backend

Deprecated since version 4.6.0: This backend has been removed in 4.6.0

- Native: Yes
- Master: No
- Slave: No
- Superslave: No
- Case: Depends
- DNSSEC: Yes, no key storage
- Disabled data: No
- Comments: No
- Zone caching: 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`.

16.12.1 Configuration Parameters

`random-hostname`

- String

Hostname for which to supply a random IP address.

16.13 Remote Backend

- Native: Yes
- Master: Yes*
- Slave: Yes*
- Superslave: Yes*
- DNSSEC: Yes*
- Zone caching: 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.

16.13.1 Important notices

There is a breaking change on v4.0 and later. Before version 4.0, the DNS names passed in queries were sent without a trailing dot, after version 4.0 the DNS names are always 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.

16.13.2 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`.

16.13.3 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
```

0MQ connector implements a REQ/REP RPC model. Please see http://zeromq.org/ for more information.

16.13.4 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

Methods required for different features

Always required `initialize`, `lookup`

Master operation `list`, `getUpdatedMasters`, `setNotified`

Slave operation `getUnfreshSlaveInfos`, `startTransaction`, `commitTransaction`, `abortTransaction`, `feedRecord`, `setFresh`

DNSSEC operation (live-signing) `getDomainKeys`, `getBeforeAndAfterNamesAbsolute`

Filling the Zone Cache `getAllDomains`

`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 domain part. Care must be taken to handle wrap-around when qname is first or last in the ordered list. Do not return nil or an empty string for `before` and `after`. When using NSEC, `unhashed` can be an empty string (but MUST be present in the response). The `qname` will be hashed when using NSEC3 and in the response, `unhashed` should be the records' real name without the domain part, and `before` and `after` should be the hashes.

- Mandatory: for NSEC/NSEC3 non-narrow
- Parameters: id (domain ID), qname
- Reply: before, after, unhashed

Example JSON/RPC

Query:

```
{"method":"getBeforeAndAfterNamesAbsolute", "params":{"id":0,"qname":"www."}}
```

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 an empty set.

- Mandatory: yes
- 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 an empty array.

- 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 and published are 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, published, content`

Example JSON/RPC

Query:

```
{"method":"getDomainKeys","parameters":{"name":"example.com."}}
```

Response:

```
{"result":[{"id":1,"flags":256,"active":true,"published":true,"content":"Private-
↪key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:␣
↪r+vmQll38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↪6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jjO4jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↪E=
Prime1: 6a165cIC0nNsGlTW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd/
↪Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↪9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↪Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪3Baj5NAi9w84l0u07KgKQZX4g0N8qUyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEUlZhf96m8WiaE1/
↪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,"published":true,"content":"Private-
↪key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:␣
↪r+vmQll38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↪6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jjO4jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↪E=
Prime1: 6a165cIC0nNsGlTW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd/
↪Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↪9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↪Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪3Baj5NAi9w84l0u07KgKQZX4g0N8qUyypnU5YDyzc6ZoagQ==
```

(continues on next page)

(continued from previous page)

```
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEUlZhf96m8WiaE1/
↪fIt4Zl2PC3fF7YIBoFLln22w=="}]}
```

addDomainKey

Adds key into local storage. See *getDomainKeys* for more information.

- Mandatory: No
- Parameters: name, key=`<flags,active,published,content>`, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"adddomainkey", "parameters":{"key":{"id":1,"flags":256,"active":true,
↪"published":true,"content":"Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:␣
↪r+vmQll38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪u7UN1oHKSKRWuZJadUwcjbp8=
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↪6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jjO4jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↪E=
Prime1: 6a165cIC0nNsGlTW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd/
↪Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↪9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↪Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪3Baj5NAi9w84l0u07KgKQZX4g0N8qUyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEUlZhf96m8WiaE1/
↪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&published=1&content=Private-key-format: v1.2
Algorithm: 8 (RSASHA256)
Modulus:␣
↪r+vmQll38ndQqNSCx9eqRBUbSOLcH4PZFX824sGhY2NSQChqt1G4ZfndzRwgjXMUwiE7GkkqU2Vbt/
↪g4iP67V/+MYecMV9YHkCRnEzb47nBXvs9JCf8AHMCnma567GQjPECh4HevPE9wmcOfpy/
↪u7UN1oHKSKRWuZJadUwcjbp8=
```

(continues on next page)

(continued from previous page)

```
PublicExponent: AQAB
PrivateExponent: CYC93UtVnOM6wrFJZ+qA9+Yx+p5yk0CSi0Q7c+/
↪6EVMuABQ5gNyTuu0j65lU3X81bwUk2wHPx6smfgoVDRAW5jjO4jgIFV6nE4inzk5YQKycQSL8YG3Nm9GciLFya1KUXs81sH
↪E=
Prime1: 6a165cIC0nNsGlTW/s2jRu7idq5+U203iE1HzSIddmWgx5KIKE/s3I+pwfmXYRUmq+4H9ASd/
↪Yot1lSYW98szw==
Prime2: wLoCPKxxnuxDx6/
↪9IKOYz8t9ZNLY74iCeQ85koqvTctkFmB9jpOUHTU9BhecaFY2euP9CuHV7z3PLtCoO8s1MQ==
Exponent1: CuzJaiR/7UboLvL4ekEy+QYCIHpX/
↪Z6FkiHK0ZRevEJUGgCHzRqvgEBXN3Jr2WYbwL4IMShmGoxzSCn8VY9BkQ==
Exponent2: LDR9/tyu0vzuLwc20B22FzNdd5rFF2wAQTQ0yF/
↪3Baj5NAi9w84l0u07KgKQZX4g0N8qUyypnU5YDyzc6ZoagQ==
Coefficient: 6S0vhIQITWzqfQSLj+wwRzs6qCvJckHb1+SD1XpwYjSgMTEUlZhf96m8WiaE1/
↪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}
```

`publishDomainKey`

Publish key id for domain name.

- Mandatory: No
- Parameters: name, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"publishdomainkey","parameters":{"name":"example.com","id":1}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/publishdomainkey/example.com/1 HTTP/1.1
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/javascript; utf-8

{"result": true}
```

`unpublishDomainKey`

Unpublish key id for domain name.

- Mandatory: No
- Parameters: name, id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"unpublishdomainkey","parameters":{"name":"example.com","id":1}}
```

Response:

```
{"result": true}
```

Example HTTP/RPC

Query:

```
POST /dnsapi/unpublishdomainkey/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}]}}
```

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
```

(continues on next page)

(continued from previous page)

```
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._upd.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: unless the zone cache has been disabled by setting *zone-cache-refresh-interval* to `0` (not recommended for performance reasons)(since 4.5.0)
- 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}]}
```

`getUpdatedMasters`

Used to find out any updates to master domains. This is used to trigger notifications in master mode.

- Mandatory: no
- Parameters: none
- Reply: array of DomainInfo or at least the `id`, `zone`, `serial` and `notified_serial` fields

Example JSON/RPC

Query:

```
{"method": "getUpdatedMasters", "parameters": {}}
```

Response:

```
{"result":[{"id":1,"zone":"unit.test.","masters":["10.0.0.1"],"notified_serial":2,
↪"serial":2,"last_check":1464693331,"kind":"master"}]}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getUpdatedMasters 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":"master"}]}
```

getUnfreshSlaveInfos

Used to find out if slave zones need checking of the master's SOA Serial.

- Mandatory: no
- Parameters: none
- Reply: array of DomainInfo or at least the `id`, `zone`, `serial` and `last_check` fields

Example JSON/RPC

Query:

```
{"method": "getUnfreshSlaveInfos", "parameters": {}}
```

Response:

```
{"result":[{"id":1,"zone":"unit.test.","masters":["10.0.0.1"],"serial":2,"last_
↪check":1464693331,"kind":"slave"}]}
```

Example HTTP/RPC

Query:

```
GET /dnsapi/getUnfreshSlaveInfos 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"],"serial":2,"last_
↪check":1464693331,"kind":"slave"}]}
```

`setFresh`

Called when a slave freshness check succeeded. This does not indicate the zone was updated on the master.

- Mandatory: No
- Parameters: id
- Reply: true for success, false for failure

Example JSON/RPC

Query:

```
{"method":"setFresh","parameters":{"id":1}}
```

Response:

```
{"result":true}
```

Example HTTP/RPC

Query:

```
PATCH /dnsapi/setFresh/1 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}
```

16.13.5 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
     }
   ]
}
```

16.14 TinyDNS Backend

- Native: Yes
- Master: Yes
- Slave: No
- Superslave: No
- DNSSEC: No
- Zone caching: Yes
- 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.

16.14.1 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 January 1st, 2017.

`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.

16.14.2 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 racked 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!

16.14.3 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 *xfr-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.

16.14.4 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 axfr-get page for the djbdns suite.

```
*.example.com     A 198.51.100.1
www.example.com   A 198.51.100.1
```

Compiling the TinyDNS backend requires you to have tinycdb version 0.77.

CHAPTER

SEVENTEEN

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.

17.1 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 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 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
- *webserver-max-bodysize*: Maximum request/response body size in megabytes
- *webserver-connection-timeout*: Request/response timeout in seconds

17.2 Metrics Endpoint

New in version 4.4.0.

The webserver exposes a metrics-endpoint that follows the prometheus exposition-format on path `/metrics`.

The metrics listed are equivalent to the variables section on the index-page of the webserver (prefixed with `pdns_auth_` and replacing dashes with underscores).

A simple `GET` request will return a response similar to the following:

`GET /metrics`

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 12044
Content-Type: text/plain
Server: PowerDNS/0.0.19015.0.master.ge719aae4e8

# HELP pdns_auth_corrupt_packets Number of corrupt packets received
# TYPE pdns_auth_corrupt_packets counter
pdns_auth_corrupt_packets 0
# HELP pdns_auth_deferred_cache_inserts Amount of cache inserts that were deferred␣
↪because of maintenance
# TYPE pdns_auth_deferred_cache_inserts counter
pdns_auth_deferred_cache_inserts 0
# HELP pdns_auth_deferred_cache_lookup Amount of cache lookups that were deferred␣
↪because of maintenance
# TYPE pdns_auth_deferred_cache_lookup counter
pdns_auth_deferred_cache_lookup 0
# HELP pdns_auth_deferred_packetcache_inserts Amount of packet cache inserts that␣
↪were deferred because of maintenance
# TYPE pdns_auth_deferred_packetcache_inserts counter
pdns_auth_deferred_packetcache_inserts 0
# HELP pdns_auth_deferred_packetcache_lookup Amount of packet cache lookups that␣
↪were deferred because of maintenance
# TYPE pdns_auth_deferred_packetcache_lookup counter
pdns_auth_deferred_packetcache_lookup 0
# HELP pdns_auth_dnsupdate_answers DNS update packets successfully answered.
# TYPE pdns_auth_dnsupdate_answers counter
pdns_auth_dnsupdate_answers 0
# HELP pdns_auth_dnsupdate_changes DNS update changes to records in total.
# TYPE pdns_auth_dnsupdate_changes counter
pdns_auth_dnsupdate_changes 0
# HELP pdns_auth_dnsupdate_queries DNS update packets received.
# TYPE pdns_auth_dnsupdate_queries counter
pdns_auth_dnsupdate_queries 0
# HELP pdns_auth_dnsupdate_refused DNS update packets that are refused.
# TYPE pdns_auth_dnsupdate_refused counter
pdns_auth_dnsupdate_refused 0
# HELP pdns_auth_incoming_notifications NOTIFY packets received.
# TYPE pdns_auth_incoming_notifications counter
pdns_auth_incoming_notifications 0
# HELP pdns_auth_overload_drops Queries dropped because backends overloaded
# TYPE pdns_auth_overload_drops counter
pdns_auth_overload_drops 0
# HELP pdns_auth_packetcache_hit Number of hits on the packet cache
# TYPE pdns_auth_packetcache_hit counter
pdns_auth_packetcache_hit 0
# HELP pdns_auth_packetcache_miss Number of misses on the packet cache
# TYPE pdns_auth_packetcache_miss counter
pdns_auth_packetcache_miss 0
# HELP pdns_auth_packetcache_size Number of entries in the packet cache
# TYPE pdns_auth_packetcache_size gauge
pdns_auth_packetcache_size 0
# HELP pdns_auth_query_cache_hit Number of hits on the query cache
# TYPE pdns_auth_query_cache_hit counter
pdns_auth_query_cache_hit 0
# HELP pdns_auth_query_cache_miss Number of misses on the query cache
# TYPE pdns_auth_query_cache_miss counter
pdns_auth_query_cache_miss 0
# HELP pdns_auth_query_cache_size Number of entries in the query cache
# TYPE pdns_auth_query_cache_size gauge
pdns_auth_query_cache_size 0
# HELP pdns_auth_rd_queries Number of recursion desired questions
```

(continues on next page)

(continued from previous page)

```
# TYPE pdns_auth_rd_queries counter
pdns_auth_rd_queries 0
# HELP pdns_auth_recursing_answers Number of recursive answers sent out
# TYPE pdns_auth_recursing_answers counter
pdns_auth_recursing_answers 0
# HELP pdns_auth_recursing_questions Number of questions sent to recursor
# TYPE pdns_auth_recursing_questions counter
pdns_auth_recursing_questions 0
# HELP pdns_auth_recursion_unanswered Number of packets unanswered by configured␣
↪recursor
# TYPE pdns_auth_recursion_unanswered counter
pdns_auth_recursion_unanswered 0
# HELP pdns_auth_security_status Security status based on regular polling
# TYPE pdns_auth_security_status gauge
pdns_auth_security_status 0
# HELP pdns_auth_servfail_packets Number of times a server-failed packet was sent␣
↪out
# TYPE pdns_auth_servfail_packets counter
pdns_auth_servfail_packets 0
# HELP pdns_auth_signatures Number of DNSSEC signatures made
# TYPE pdns_auth_signatures counter
pdns_auth_signatures 0
# HELP pdns_auth_tcp_answers Number of answers sent out over TCP
# TYPE pdns_auth_tcp_answers counter
pdns_auth_tcp_answers 0
# HELP pdns_auth_tcp_answers_bytes Total size of answers sent out over TCP
# TYPE pdns_auth_tcp_answers_bytes counter
pdns_auth_tcp_answers_bytes 0
# HELP pdns_auth_tcp_queries Number of TCP queries received
# TYPE pdns_auth_tcp_queries counter
pdns_auth_tcp_queries 0
# HELP pdns_auth_tcp4_answers Number of IPv4 answers sent out over TCP
# TYPE pdns_auth_tcp4_answers counter
pdns_auth_tcp4_answers 0
# HELP pdns_auth_tcp4_answers_bytes Total size of answers sent out over TCPv4
# TYPE pdns_auth_tcp4_answers_bytes counter
pdns_auth_tcp4_answers_bytes 0
# HELP pdns_auth_tcp4_queries Number of IPv4 TCP queries received
# TYPE pdns_auth_tcp4_queries counter
pdns_auth_tcp4_queries 0
# HELP pdns_auth_tcp6_answers Number of IPv6 answers sent out over TCP
# TYPE pdns_auth_tcp6_answers counter
pdns_auth_tcp6_answers 0
# HELP pdns_auth_tcp6_answers_bytes Total size of answers sent out over TCPv6
# TYPE pdns_auth_tcp6_answers_bytes counter
pdns_auth_tcp6_answers_bytes 0
# HELP pdns_auth_tcp6_queries Number of IPv6 TCP queries received
# TYPE pdns_auth_tcp6_queries counter
pdns_auth_tcp6_queries 0
# HELP pdns_auth_timedout_packets Number of packets which weren't answered within␣
↪timeout set
# TYPE pdns_auth_timedout_packets counter
pdns_auth_timedout_packets 0
# HELP pdns_auth_udp_answers Number of answers sent out over UDP
# TYPE pdns_auth_udp_answers counter
pdns_auth_udp_answers 0
# HELP pdns_auth_udp_answers_bytes Total size of answers sent out over UDP
# TYPE pdns_auth_udp_answers_bytes counter
pdns_auth_udp_answers_bytes 0
# HELP pdns_auth_udp_do_queries Number of UDP queries received with DO bit
# TYPE pdns_auth_udp_do_queries counter
```

(continues on next page)

(continued from previous page)

```
pdns_auth_udp_do_queries 0
# HELP pdns_auth_udp_queries Number of UDP queries received
# TYPE pdns_auth_udp_queries counter
pdns_auth_udp_queries 0
# HELP pdns_auth_udp4_answers Number of IPv4 answers sent out over UDP
# TYPE pdns_auth_udp4_answers counter
pdns_auth_udp4_answers 0
# HELP pdns_auth_udp4_answers_bytes Total size of answers sent out over UDPv4
# TYPE pdns_auth_udp4_answers_bytes counter
pdns_auth_udp4_answers_bytes 0
# HELP pdns_auth_udp4_queries Number of IPv4 UDP queries received
# TYPE pdns_auth_udp4_queries counter
pdns_auth_udp4_queries 0
# HELP pdns_auth_udp6_answers Number of IPv6 answers sent out over UDP
# TYPE pdns_auth_udp6_answers counter
pdns_auth_udp6_answers 0
# HELP pdns_auth_udp6_answers_bytes Total size of answers sent out over UDPv6
# TYPE pdns_auth_udp6_answers_bytes counter
pdns_auth_udp6_answers_bytes 0
# HELP pdns_auth_udp6_queries Number of IPv6 UDP queries received
# TYPE pdns_auth_udp6_queries counter
pdns_auth_udp6_queries 0
# HELP pdns_auth_cpu_iowait Time spent waiting for I/O to complete by the whole␣
↪system, in units of USER_HZ
# TYPE pdns_auth_cpu_iowait counter
pdns_auth_cpu_iowait 2739
# HELP pdns_auth_cpu_steal Stolen time, which is the time spent by the whole␣
↪system in other operating systems when running in a virtualized environment, in␣
↪units of USER_HZ
# TYPE pdns_auth_cpu_steal counter
pdns_auth_cpu_steal 0
# HELP pdns_auth_fd_usage Number of open filedescriptors
# TYPE pdns_auth_fd_usage gauge
pdns_auth_fd_usage 26
# HELP pdns_auth_key_cache_size Number of entries in the key cache
# TYPE pdns_auth_key_cache_size gauge
pdns_auth_key_cache_size 0
# HELP pdns_auth_latency Average number of microseconds needed to answer a question
# TYPE pdns_auth_latency gauge
pdns_auth_latency 0
# HELP pdns_auth_meta_cache_size Number of entries in the metadata cache
# TYPE pdns_auth_meta_cache_size gauge
pdns_auth_meta_cache_size 0
# HELP pdns_auth_open_tcp_connections Number of currently open TCP connections
# TYPE pdns_auth_open_tcp_connections gauge
pdns_auth_open_tcp_connections 0
# HELP pdns_auth_qsize_q Number of questions waiting for database attention
# TYPE pdns_auth_qsize_q gauge
pdns_auth_qsize_q 0
# HELP pdns_auth_real_memory_usage Actual unique use of memory in bytes (approx)
# TYPE pdns_auth_real_memory_usage gauge
pdns_auth_real_memory_usage 133189632
# HELP pdns_auth_ring_logmessages_capacity Maximum number of entries in the␣
↪logmessages ring
# TYPE pdns_auth_ring_logmessages_capacity gauge
pdns_auth_ring_logmessages_capacity 10000
# HELP pdns_auth_ring_logmessages_size Number of entries in the logmessages ring
# TYPE pdns_auth_ring_logmessages_size gauge
pdns_auth_ring_logmessages_size 7
# HELP pdns_auth_ring_noerror_queries_capacity Maximum number of entries in the␣
↪noerror-queries ring
```

(continues on next page)

(continued from previous page)

```
# TYPE pdns_auth_ring_noerror_queries_capacity gauge
pdns_auth_ring_noerror_queries_capacity 10000
# HELP pdns_auth_ring_noerror_queries_size Number of entries in the noerror-
↪queries ring
# TYPE pdns_auth_ring_noerror_queries_size gauge
pdns_auth_ring_noerror_queries_size 0
# HELP pdns_auth_ring_nxdomain_queries_capacity Maximum number of entries in the␣
↪nxdomain-queries ring
# TYPE pdns_auth_ring_nxdomain_queries_capacity gauge
pdns_auth_ring_nxdomain_queries_capacity 10000
# HELP pdns_auth_ring_nxdomain_queries_size Number of entries in the nxdomain-
↪queries ring
# TYPE pdns_auth_ring_nxdomain_queries_size gauge
pdns_auth_ring_nxdomain_queries_size 0
# HELP pdns_auth_ring_queries_capacity Maximum number of entries in the queries␣
↪ring
# TYPE pdns_auth_ring_queries_capacity gauge
pdns_auth_ring_queries_capacity 10000
# HELP pdns_auth_ring_queries_size Number of entries in the queries ring
# TYPE pdns_auth_ring_queries_size gauge
pdns_auth_ring_queries_size 0
# HELP pdns_auth_ring_remotes_capacity Maximum number of entries in the remotes␣
↪ring
# TYPE pdns_auth_ring_remotes_capacity gauge
pdns_auth_ring_remotes_capacity 10000
# HELP pdns_auth_ring_remotes_corrupt_capacity Maximum number of entries in the␣
↪remotes-corrupt ring
# TYPE pdns_auth_ring_remotes_corrupt_capacity gauge
pdns_auth_ring_remotes_corrupt_capacity 10000
# HELP pdns_auth_ring_remotes_corrupt_size Number of entries in the remotes-
↪corrupt ring
# TYPE pdns_auth_ring_remotes_corrupt_size gauge
pdns_auth_ring_remotes_corrupt_size 0
# HELP pdns_auth_ring_remotes_size Number of entries in the remotes ring
# TYPE pdns_auth_ring_remotes_size gauge
pdns_auth_ring_remotes_size 0
# HELP pdns_auth_ring_remotes_unauth_capacity Maximum number of entries in the␣
↪remotes-unauth ring
# TYPE pdns_auth_ring_remotes_unauth_capacity gauge
pdns_auth_ring_remotes_unauth_capacity 10000
# HELP pdns_auth_ring_remotes_unauth_size Number of entries in the remotes-unauth␣
↪ring
# TYPE pdns_auth_ring_remotes_unauth_size gauge
pdns_auth_ring_remotes_unauth_size 0
# HELP pdns_auth_ring_servfail_queries_capacity Maximum number of entries in the␣
↪servfail-queries ring
# TYPE pdns_auth_ring_servfail_queries_capacity gauge
pdns_auth_ring_servfail_queries_capacity 10000
# HELP pdns_auth_ring_servfail_queries_size Number of entries in the servfail-
↪queries ring
# TYPE pdns_auth_ring_servfail_queries_size gauge
pdns_auth_ring_servfail_queries_size 0
# HELP pdns_auth_ring_unauth_queries_capacity Maximum number of entries in the␣
↪unauth-queries ring
# TYPE pdns_auth_ring_unauth_queries_capacity gauge
pdns_auth_ring_unauth_queries_capacity 10000
# HELP pdns_auth_ring_unauth_queries_size Number of entries in the unauth-queries␣
↪ring
# TYPE pdns_auth_ring_unauth_queries_size gauge
pdns_auth_ring_unauth_queries_size 0
# HELP pdns_auth_signature_cache_size Number of entries in the signature cache
```

(continues on next page)

(continued from previous page)

```
# TYPE pdns_auth_signature_cache_size gauge
pdns_auth_signature_cache_size 0
# HELP pdns_auth_sys_msec Number of msec spent in system time
# TYPE pdns_auth_sys_msec counter
pdns_auth_sys_msec 56
# HELP pdns_auth_udp_in_errors UDP 'in' errors
# TYPE pdns_auth_udp_in_errors counter
pdns_auth_udp_in_errors 151
# HELP pdns_auth_udp_noport_errors UDP 'noport' errors
# TYPE pdns_auth_udp_noport_errors counter
pdns_auth_udp_noport_errors 9
# HELP pdns_auth_udp_recvbuf_errors UDP 'recvbuf' errors
# TYPE pdns_auth_udp_recvbuf_errors counter
pdns_auth_udp_recvbuf_errors 0
# HELP pdns_auth_udp_sndbuf_errors UDP 'sndbuf' errors
# TYPE pdns_auth_udp_sndbuf_errors counter
pdns_auth_udp_sndbuf_errors 9
# HELP pdns_auth_uptime Uptime of process in seconds
# TYPE pdns_auth_uptime counter
pdns_auth_uptime 672
# HELP pdns_auth_user_msec Number of msec spent in user time
# TYPE pdns_auth_user_msec counter
pdns_auth_user_msec 48
```

Prometheus can then be configured to scrape metrics from this endpoint using a simple job description like the following:

```
scrape_configs:
  - job_name: 'pdns_auth'
    scrape_interval: 1m
    static_configs:
      - targets: ['pdns_auth_host:pdns_auth_ws_port']
```

Further details can be gathered from the prometheus docs.

17.3 Enabling the API

To enable the API, the webserver and the HTTP API need to be enabled. Add these lines to `pdns.conf`:

```
api=yes
api-key=changeme
```

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 .
```

17.4 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. It can also be obtained from a running server if the administrator of that server has enabled the API; it is available at the */api/docs* endpoint in both YAML and JSON formats (the 'Accept' header can be used to indicate the desired format).

17.4.1 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.

17.4.2 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 is not a hash: `400 Bad Request`

Error responses have a JSON body of this format:

`Error`

Returned when the server encounters an error, either in client input or internally

Object Properties

- **`error`** (*`string`*) – A human readable error message
- **`errors`** (*`[string]`*) – Optional array of multiple errors encountered during processing

17.4.3 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 encoded in UTF-8 and HTTP headers will reflect this.

Data types:

- empty fields: `null` but present
- Regex: implementation defined
- Dates: ISO 8601

17.5 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:

17.5.1 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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* 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

Examples

Listing all servers

```
GET /api/v1/servers HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

[{"autoprimaries_url": "/api/v1/servers/localhost/autoprimaries{/autoprimary}",
↪"config_url": "/api/v1/servers/localhost/config{/config_setting}", "daemon_type
↪": "authoritative", "id": "localhost", "type": "Server", "url": "/api/v1/servers/
↪localhost", "version": "4.6.1", "zones_url": "/api/v1/servers/localhost/zones{/
↪zone}"}]
```

(continues on next page)

(continued from previous page)

Listing a server

```
GET /api/v1/servers/localhost HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

{"autoprimaries_url": "/api/v1/servers/localhost/autoprimaries{/autoprimary}", "config_url": "/api/v1/servers/localhost/config{/config_setting}", "daemon_type": "authoritative", "id": "localhost", "type": "Server", "url": "/api/v1/servers/localhost", "version": "4.6.1", "zones_url": "/api/v1/servers/localhost/zones{/zone}"}
```

17.5.2 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

Query Parameters

- **zone** (*string*) – When set to the name of a zone, only this zone is returned. If no zone with that name exists, the response is an empty array. This can e.g. be used to check if a zone exists in the database without having to guess/encode the zone's id or to check if a zone exists.
- **dnssec** (*boolean*) – "true" (default) or "false", whether to include the "dnssec" and "edited_serial" fields in the Zone objects. Setting this to "false" will make the query a lot faster.

Status Codes

- 200 OK – An array of Zones Returns: array of *Zone* objects
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* 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

Query Parameters

- **rrsets** (*boolean*) – "true" (default) or "false", whether to include the "rrsets" in the response Zone object.
- **rrset_name** (*string*) – Limit output to RRsets for this name.
- **rrset_type** (*string*) – Limit output to the RRset of this type. Can only be used together with rrset_name.

Status Codes

- 200 OK – A Zone Returns: *Zone* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* 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

- 204 No Content – Returns 204 No Content on success.
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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.
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

PUT /servers/{server_id}/zones/{zone_id}

Modifies basic zone data.

The only fields in the zone structure which can be modified are: kind, masters, catalog, account, soa_edit, soa_edit_api, api_rectify, dnssec, and nsec3param. All other fields are ignored.

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.
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

PUT /servers/{server_id}/zones/{zone_id}/axfr-retrieve

Retrieve slave zone from its master.

Fails when zone kind is not Slave, or slave is disabled in the configuration. 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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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", "Producer", "Consumer"
- **rrsets** ([*RRSet*]) – RRSets in this zone (for zones/{zone_id} endpoint only; omitted during GET on the …/zones list endpoint)
- **serial** (*integer*) – The SOA serial number
- **notified_serial** (*integer*) – The SOA serial notifications have been sent out for
- **edited_serial** (*integer*) – The SOA serial as seen in query responses. Calculated using the SOA-EDIT metadata, default-soa-edit and default-soa-edit-signed settings
- **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
- **catalog** (*string*) – The catalog this zone is a member of
- **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.
- **master_tsig_key_ids** (*[string]*) – The id of the TSIG keys used for master operation in this zone
- **slave_tsig_key_ids** (*[string]*) – The id of the TSIG keys used for slave operation in this zone

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")
- **ttl** (*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. When unset, the record is not disabled

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`.

Examples

Listing all zones

```
GET /api/v1/servers/localhost/zones HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

[{"account": "", "dnssec": false, "edited_serial": 2022040504, "id": "example.org.
↪", "kind": "Native", "last_check": 0, "masters": [], "name": "example.org.",
↪"notified_serial": 0, "serial": 2022040504, "url": "/api/v1/servers/localhost/
↪zones/example.org."}]
```

Creating new zone

```
POST /api/v1/servers/localhost/zones HTTP/1.1
X-API-Key: secret
Content-Type: application/json

{"name": "example.org.", "kind": "Native", "masters": [], "nameservers": ["ns1.
↪example.org.", "ns2.example.org."]}
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

{"account": "", "api_rectify": false, "dnssec": false, "edited_serial": 2022040501,
↪ "id": "example.org.", "kind": "Native", "last_check": 0, "master_tsig_key_ids":␣
↪[], "masters": [], "name": "example.org.", "notified_serial": 0, "nsec3narrow":␣
↪false, "nsec3param": "", "rrsets": [{"comments": [], "name": "example.org.",
↪"records": [{"content": "a.misconfigured.dns.server.invalid. hostmaster.example.
↪org. 2022040501 10800 3600 604800 3600", "disabled": false}], "ttl": 3600, "type
↪": "SOA"}, {"comments": [], "name": "example.org.", "records": [{"content": "ns1.
↪example.org.", "disabled": false}, {"content": "ns2.example.org.", "disabled":␣
↪false}], "ttl": 3600, "type": "NS"}], "serial": 2022040501, "slave_tsig_key_ids
↪": [], "soa_edit": "", "soa_edit_api": "DEFAULT", "url": "/api/v1/servers/
↪localhost/zones/example.org."}
```

Listing a zone

```
GET /api/v1/servers/localhost/zones/example.org. HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

{"account": "", "api_rectify": false, "dnssec": false, "edited_serial": 2022040501,
↪ "id": "example.org.", "kind": "Native", "last_check": 0, "master_tsig_key_ids":␣
↪[], "masters": [], "name": "example.org.", "notified_serial": 0, "nsec3narrow":␣
↪false, "nsec3param": "", "rrsets": [{"comments": [], "name": "example.org.",
↪"records": [{"content": "a.misconfigured.dns.server.invalid. hostmaster.example.
↪org. 2022040501 10800 3600 604800 3600", "disabled": false}], "ttl": 3600, "type
↪": "SOA"}, {"comments": [], "name": "example.org.", "records": [{"content": "ns1.
↪example.org.", "disabled": false}, {"content": "ns2.example.org.", "disabled":␣
↪false}], "ttl": 3600, "type": "NS"}], "serial": 2022040501, "slave_tsig_key_ids
```

(continues on next page)

(continued from previous page)

Deleting a zone

```
DELETE /api/v1/servers/localhost/zones/example.org. HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 204 No Content
```

Creating new RRset

```
PATCH /api/v1/servers/localhost/zones/example.org. HTTP/1.1
X-API-Key: secret
Content-Type: application/json

{"rrsets": [{"name": "test.example.org.", "type": "A", "ttl": 3600, "changetype": "REPLACE", "records": [{"content": "192.168.0.5", "disabled": false}]}]}
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 204 No Content
```

Deleting a RRset

```
PATCH /api/v1/servers/localhost/zones/example.org. HTTP/1.1
X-API-Key: secret
Content-Type: application/json

{"rrsets": [{"name": "test.example.org.", "type": "A", "changetype": "DELETE"}]}
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 204 No Content
```

Rectifying a zone

```
PUT /api/v1/servers/localhost/zones/example.org./rectify HTTP/1.1
X-API-Key: secret
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "Rectified"}
```

17.5.3 Cryptokeys

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* 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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* 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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

Cryptokey

Describes a DNSSEC cryptographic key

Object Properties

- **type** (*string*) – set to "Cryptokey"
- **id** (*integer*) – The internal identifier, read only
- **keytype** (*string*) –
- **active** (*boolean*) – Whether or not the key is in active use
- **published** (*boolean*) – Whether or not the DNSKEY record is published in the zone
- **dnskey** (*string*) – The DNSKEY record for this key
- **ds** (*[string]*) – An array of DS records for this key
- **cds** (*[string]*) – An array of DS records for this key, filtered by CDS publication settings

- **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

17.5.4 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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

GET /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}
Get the content of a single kind of domain metadata as a Metadata object.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) – The id of the zone to retrieve
- **metadata_kind** (*string*) – The kind of metadata

Status Codes

- 200 OK – Metadata object with list of values Returns: *Metadata* object

- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

PUT /servers/{server_id}/zones/{zone_id}/metadata/{metadata_kind}

Replace the content of a single kind of domain metadata.

Creates a set of metadata entries of given kind for the zone. Existing metadata entries for the zone with the same kind are removed.

Parameters

- **server_id** (*string*) – The id of the server to retrieve
- **zone_id** (*string*) –
- **metadata_kind** (*string*) – The kind of metadata

Status Codes

- 200 OK – Metadata object with list of values Returns: *Metadata* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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*) – The kind of metadata

Status Codes

- 204 No Content – OK
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

Metadata

Represents zone metadata

Object Properties

- **kind** (*string*) – Name of the metadata
- **metadata** (*[string]*) – Array with all values for this metadata kind.

17.5.5 TSIGKeys

New in version 4.2.0.

TSIGKeys can be manipulated via the API.

TSIGKey Endpoints

GET /servers/{server_id}/tsigkeys
Get all TSIGKeys on the server, except the actual key

Parameters

- **server_id** (*string*) – The id of the server

Status Codes

- 200 OK – List of TSIGKey objects Returns: array of *TSIGKey* objects
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

POST /servers/{server_id}/tsigkeys
Add a TSIG key

This methods add a new TSIGKey. The actual key can be generated by the server or be provided by the client

Parameters

- **server_id** (*string*) – The id of the server

Status Codes

- 201 Created – Created Returns: *TSIGKey* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 409 Conflict – An item with this name already exists Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

GET /servers/{server_id}/tsigkeys/{tsigkey_id}
Get a specific TSIGKeys on the server, including the actual key

Parameters

- **server_id** (*string*) – The id of the server to retrieve the key from
- **tsigkey_id** (*string*) – The id of the TSIGkey. Should match the "id" field in the TSIGKey object

Status Codes

- 200 OK – OK. Returns: *TSIGKey* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object

- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

PUT /servers/{server_id}/tsigkeys/{tsigkey_id}

The TSIGKey at tsigkey_id can be changed in multiple ways:

- Changing the Name, this will remove the key with tsigkey_id after adding.
- Changing the Algorithm
- Changing the Key

Only the relevant fields have to be provided in the request body.

Parameters

- **server_id** (*string*) – The id of the server to retrieve the key from
- **tsigkey_id** (*string*) – The id of the TSIGkey. Should match the "id" field in the TSIGKey object

Status Codes

- 200 OK – OK. TSIGKey is changed. Returns: *TSIGKey* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 409 Conflict – An item with this name already exists Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

DELETE /servers/{server_id}/tsigkeys/{tsigkey_id}

Delete the TSIGKey with tsigkey_id

Parameters

- **server_id** (*string*) – The id of the server to retrieve the key from
- **tsigkey_id** (*string*) – The id of the TSIGkey. Should match the "id" field in the TSIGKey object

Status Codes

- 204 No Content – OK, key was deleted
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

TSIGKey

A TSIG key that can be used to authenticate NOTIFY, AXFR, and DNSUPDATE queries.

Object Properties

- **name** (*string*) – The name of the key
- **id** (*string*) – The ID for this key, used in the TSIGkey URL endpoint.

- **algorithm** (*string*) – The algorithm of the TSIG key
- **key** (*string*) – The Base64 encoded secret key, empty when listing keys. MAY be empty when POSTing to have the server generate the key material
- **type** (*string*) – Set to "TSIGKey"

Examples

Generating a new TSIG key

```
POST /servers/localhost/tsigkeys HTTP/1.1
X-Api-Key: secret
Content-Type: application/json

{"name": "mytsigkey", "algorithm": "hmac-sha256"}
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 201 Created
Content-Type: application/json

{"algorithm": "hmac-sha256", "id": "mytsigkey.", "key":
↪"ayZmdUpT5xh7H21p3UUHJWZgF2F2jNncqx5SQkRIWbqReuwieiVBG8jtEieb/
↪njCbjrLtQkjVsgoiKCtsqNsdQ==", "name": "mytsigkey", "type": "TSIGKey"}
```

Modifying the key material

```
PUT /servers/localhost/tsigkeys/mytsigkey. HTTP/1.1
X-Api-Key: secret
Content-Type: application/json

{"name": "mytsigkey", "key":
↪"GQNyFy1QagMUarHmiSgsIJajghdTGJGVcN5TRVwgbclzxGyhQR1uYLCOyJ/
↪uj9uj12jyeLwzJuW12wCI9PYv7Q=="}
```

```
HTTP/1.1 200 Ok
Content-Type: application/json

{"algorithm": "hmac-sha256", "id": "mytsigkey.", "key":
↪"GQNyFy1QagMUarHmiSgsIJajghdTGJGVcN5TRVwgbclzxGyhQR1uYLCOyJ/
↪uj9uj12jyeLwzJuW12wCI9PYv7Q==", "name": "mytsigkey", "type": "TSIGKey"}
```

17.5.6 Autoprimaries

This API is used to manage *autoprimaries*.

Autoprimary endpoints

GET /servers/{server_id}/autoprimaries

Get a list of autoprimaries

Parameters

- **server_id** (*string*) – The id of the server to manage the list of autoprimaries on

Status Codes

- 200 OK – OK. Returns: `Autoprimary server` object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

POST /servers/{server_id}/autoprimaries

Add an autoprimary

This methods add a new autoprimary server.

Parameters

- **server_id** (*string*) – The id of the server to manage the list of autoprimaries on

Status Codes

- 201 Created – Created
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

DELETE /servers/{server_id}/autoprimaries/{ip}/{nameserver}

Delete the autoprimary entry

Parameters

- **server_id** (*string*) – The id of the server to delete the autoprimary from
- **ip** (*string*) – IP address of autoprimary
- **nameserver** (*string*) – DNS name of the autoprimary

Status Codes

- 204 No Content – OK, key was deleted
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

An autoprimary object represents a single autoprimary server.

Autoprimary

An autoprimary server that can provision new domains.

Object Properties

- **ip** (*string*) – IP address of the autoprimary server
- **nameserver** (*string*) – DNS name of the autoprimary server
- **account** (*string*) – Account name for the autoprimary server

Examples

Listing autoprimaries

```
GET /servers/localhost/autoprimaries HTTP/1.1
X-Api-Key: secret
Content-Type: application/json
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 200 Ok
Content-Type: application/json

[{"ip":"192.0.2.1","nameserver":"ns.example.com","account":""},{"ip":"192.0.2.50",
↪"nameserver":"ns.example.org","account":"example"}]
```

Creating an autoprimary

```
POST /servers/localhost/autoprimaries HTTP/1.1
X-Api-Key: secret
Content-Type: application/json

{"ip":"192.0.2.1","nameserver":"ns.example.com","account":""}
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 201 Created
```

Deleting an autoprimary

```
DELETE /servers/localhost/autoprimaries/192.0.2.1/ns.example.com HTTP/1.1
X-Api-Key: secret
Content-Type: application/json
```

Will yield a response similar to this (several headers omitted):

```
HTTP/1.1 204 No Content
```

17.5.7 Searching

The API allows searching for data in *Zone*s, *Comment*s and *RRSet*s.

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
- **object_type** (*string*) – Type of data to search for, one of "all", "zone", "record", "comment"

Status Codes

- 200 OK – Returns a JSON array with results Returns: array of *SearchResult* objects
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

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*) –

17.5.8 Statistics

Endpoints

GET /servers/{server_id}/statistics

Query statistics.

Query PowerDNS internal statistics.

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Query Parameters

- **statistic** (*string*) – When set to the name of a specific statistic, only this value is returned. If no statistic with that name exists, the response has a 422 status and an error message.
- **includerings** (*boolean*) – "true" (default) or "false", whether to include the Ring items, which can contain thousands of log messages or queried domains. Setting this to "false" may make the response a lot smaller.

Status Codes

- 200 OK – List of Statistic Items Returns: array

- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – Returned when a non-existing statistic name has been requested. Contains an error message
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

The Statistics endpoint returns an array of objects that can be StatisticItem, MapStatisticItem or RingStatisticItem :

StatisticItem

Object Properties

- **name** (*string*) – Item name
- **type** (*string*) – set to "StatisticItem"
- **value** (*string*) – Item value

MapStatisticItem

Object Properties

- **name** (*string*) – Item name
- **type** (*string*) – Set to "MapStatisticItem"
- **value** ([*SimpleStatisticItem*]) – Named values

RingStatisticItem

Object Properties

- **name** (*string*) – Item name
- **type** (*string*) – Set to "RingStatisticItem"
- **size** (*integer*) – Ring size
- **value** ([*SimpleStatisticItem*]) – Named values

Both MapStatisticItem and RingStatisticItem objects contains an array of SimpleStatisticItem

SimpleStatisticItem

Object Properties

- **name** (*string*) – Item name
- **value** (*string*) – Item value

17.5.9 Cache

Endpoints

PUT /servers/{server_id}/cache/flush

Flush a cache-entry by name

Parameters

- **server_id** (*string*) – The id of the server to retrieve

Query Parameters

- **domain** (*string*) – The domain name to flush from the cache

Status Codes

- 200 OK – Flush successful Returns: *CacheFlushResult* object
- 400 Bad Request – The supplied request was not valid Returns: *Error* object
- 404 Not Found – Requested item was not found Returns: *Error* object
- 422 Unprocessable Entity – The input to the operation was not valid Returns: *Error* object
- 500 Internal Server Error – Internal server error Returns: *Error* object

Objects

CacheFlushResult

The result of a cache-flush

Object Properties

- **count** (*number*) – Amount of entries flushed
- **result** (*string*) – A message about the result like "Flushed cache"

CHAPTER

EIGHTEEN

MANUAL PAGES

The PowerDNS Authoritative Server comes with many binaries. The manual pages for these programs are included here:

18.1 calidns

18.1.1 Synopsis

`calidns` [*OPTIONS*] *QUERY_FILE DESTINATION INITIAL_QPS HITRATE*

18.1.2 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*.

18.1.3 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.

18.1.4 Options

--ecs <SUBNET> Add EDNS Client Subnet option to outgoing queries using random addresses from the specified *SUBNET* range (IPv4 only).

--ecs-from-file Read IP or subnet values from the query file and add them as EDNS Client Subnet option to outgoing queries.

--increment <NUM> On every subsequent run, multiply the number of queries per second by *NUM*. By default, this is 1.1.

--maximum-qps <NUM> Stop incrementing once this rate has been reached, to provide a stable load.

--minimum-success-rate <NUM> Stop the test as soon as the success rate drops below this value, in percent.

--plot-file <FILE> Write results to the specified file.

--quiet Whether to run quietly, outputting only the maximum QPS reached. This option is mostly useful when used with `--minimum-success-rate`.

--want-recursion Set this flag to send queries with the Recursion Desired flag set.

18.2 dnsbulktest

18.2.1 Synopsis

`dnsbulktest` [*OPTION*]. . . *IPADDRESS PORT* [*LIMIT*]

18.2.2 Description

`dnsbulktest` 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.

18.2.3 Options

--help, -h Show a summary of options.

--quiet, -q Don't show information on individual queries.

--type, -t <TYPE> Query the nameserver for *TYPE*, A by default.

--envoutput, -e Write results on STDOUT as shell environment variables

--version Display the version of dnsbulktest

18.3 dnsgram

18.3.1 Synopsis

`dnsgram` *INFILE. . .*

18.3.2 Description

`dnsgram` takes one or more *INFILE*s in PCAP format and generates statistics on 5 second segments allowing the study of intermittent resolver issues.

18.3.3 Options

None

18.3.4 See also

pcap(3PCAP), tcpdump(8)

18.4 dnspcap2calidns

18.4.1 Synopsis

`dnspcap2calidns` *PCAPFILE OUTFILE*

18.4.2 Description

`dnspcap2calidns` reads the PCAP file *PCAPFILE* for DNS queries and writes these to *OUTFILE* in the format understood by **`calidns`**.

18.4.3 Options

--help	Show a summary of options.
--version	Display the version of dnspcap2calidns

18.4.4 See also

`calidns(1)`

18.5 dnspcap2protobuf

18.5.1 Synopsis

`dnspcap2protobuf` *PCAPFILE OUTFILE*

18.5.2 Description

`dnspcap2protobuf` reads the PCAP file *PCAPFILE* for DNS queries and responses and writes these in the PowerDNS protobuf format to *OUTFILE*.

18.5.3 Options

--help	Show a summary of options.
--version	Display the version of dnspcap2protobuf

18.6 dnsreplay

18.6.1 Synopsis

`dnsreplay` [*OPTION*]... *FILENAME ADDRESS* [*PORT*]

18.6.2 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 will only replay queries with recursion-desired flag set.

18.6.3 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

--packet-limit <NUM> Stop after replaying *NUM* packets. Default for *NUM* is 0, which means no limit.

--pcap-dns-port <VAL> Look at packets from or to this port in the PCAP. Default is 53.

--quiet <FLAG> If *FLAG* is set to 1, **dnsreplay** will not be very noisy with its output. This is the default.

--recursive <FLAG> If *FLAG* is set to 1, **dnsreplay** will only replay queries with recursion desired flag set. This is the default.

--source-from-pcap <FLAG> If *FLAG* is set to 1, **dnsreplay** will send the replayed queries from the source IP address and port present in the PCAP file. This requires IP_TRANSPARENT support. Default is 0 which means replayed queries will be sent from a local address.

--source-ip <VAL> Send the replayed queries from the source IP specified in *VAL*. Default is to send them from a local address.

--source-port <VAL> Send the replayed queries from the source port specified in *VAL*. Default is to send from a random port selected by the kernel.

--speedup <FACTOR> Replay queries with this speedup *FACTOR*. Default is 1.

--timeout-msec <MSEC> Wait at least *MSEC* milliseconds for a reply. Default is 500.

18.6.4 Bugs

dnsreplay has no certain handling for timeouts. It handles around at most 65536 outstanding answers.

18.6.5 See also

pcap(3PCAP), tcpdump(8), dnswasher(1)

18.7 dnsscan

18.7.1 Synopsis

`dnsscan` *INFILE…*

18.7.2 Description

`dnsscan` takes one or more *INFILE*s in PCAP format and generates a list of the number of queries per qtype.

18.7.3 Options

None

18.7.4 See also

pcap(3PCAP), tcpdump(8)

18.8 dnsscope

18.8.1 Synopsis

`dnsscope` [*OPTION*]… *INFILE*

18.8.2 Description

`dnsscope` takes an *INFILE* in PCAP format. It generates some simple statistics outputs these to STDOUT.

18.8.3 Options

INFILE Path to a PCAP file.

-h, --help Show the help.

--rd Only process packets in *INFILE* with the RD (Recursion Desired) flag set. By default, we process all DNS packets in *INFILE*.

--ipv4=<state> Process IPv4 packets. On by default, disable with **–ipv4 false**.

--ipv6=<state> Process IPv6 packets. On by default, disable with **–ipv6 false**.

-f, --filter-name=<domain> Only process packets within this domain

--full-histogram <msec> Write out histogram with specified bin-size to 'full-histogram'

--log-histogram Write out a log-histogram of response times to 'log-histogram'

--no-servfail-stats Remove servfail responses from latency statistics

--port The source and destination port to consider. Default is looking at packets from and to ports 53 and 5300.

--servfail-tree Figure out subtrees that generate servfails.

--stats-dir <directory> Drop statistics files in this directory. Defaults to ./

-l, --load-stats Emit per-second load statistics (questions, answers, outstanding).

-w <file>, --write-failures <file> Write weird packets to a PCAP file at *FILENAME*.

-v, --verbose Be more verbose.

18.8.4 See also

pcap(3PCAP), tcpdump(8)

18.9 dnstcpbench

18.9.1 Synopsis

dnstcpbench [*OPTION*]... *REMOTE-ADDRESS* [*REMOTE-PORT*]

18.9.2 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.

18.9.3 Options

REMOTE-ADDRESS IPv4 or IPv6 to test against.

REMOTE-PORT Port to test against, defaults to 53.

-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.

18.9.4 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.

18.10 dnswasher

18.10.1 Synopsis

`dnswasher` [*OPTION*]... *INFILE* [*INFILE*] *OUTFILE*

18.10.2 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.

18.10.3 Options

--decrypt, -d	Undo IPCipher encryption of IP addresses.
--help, -h	Show summary of options.
--key, -k	Base64 encoded 128-bit key for IPCipher.
--passphrase, -p	Passphrase that will be used to derive an IPCipher key.
--version, -v	Output version.

18.10.4 See also

pcap(3PCAP), tcpdump(8)

18.11 dumresp

18.11.1 Synopsis

dumresp *LOCAL-ADDRESS LOCAL-PORT NUMBER-OF-PROCESSES* [tcp]

18.11.2 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.

18.11.3 Options

tcp: Whether to listen and accept TCP connections in addition to UDP packets. Defaults to false.

18.11.4 See also

socket(7)

18.12 ixfrdist

18.12.1 Synopsis

ixfrdist [*OPTION*]...

18.12.2 Description

ixfrdist transfers zones from an authoritative server and re-serves these zones over AXFR and IXFR. It checks the SOA serial for all configured domains and downloads new versions to disk.

When a SOA query comes in and the client's address is allowed by the ACL, **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 to the client. Should an IXFR be served, **ixfrdist** will condense all differences it has for the domain into one IXFR.

ixfrdist is configured with a configuration file in YAML format. Please see `ixfrdist.yml(5)` for information.

18.12.3 Options

--help Show all supported options

--verbose Log INFO messages

--debug Log INFO and DEBUG messages

--version Display the version of ixfrdist

--config <PATH> Load configuration from *PATH*, a YAML file. When not set, an *ixfrdist.yml* is attempted to be read from the SYSCONFDIR.

18.12.4 See also

ixplore(1), pdns_server(1), ixfrdist.yml(5)

18.13 ixplore

18.13.1 Synopsis

ixplore *COMMAND COMMAND_OPT*...

ixplore diff *ZONE BEFORE AFTER*

ixplore track *IP ADDRESS PORT ZONE DIRECTORY*

18.13.2 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.

18.13.3 Options

18.13.4 diff-mode

ZONE The name of the zone the IXFRs are consumed from.

BEFORE Path to the 'before' zonefile.

AFTER Path to the 'after' zonefile.

18.13.5 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.

18.13.6 See also

diff(1)

18.14 nproxy

18.14.1 Synopsis

nproxy –powerdns-address *ADDRESS* [*OPTION*]… *ADDRESS*…

18.14.2 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.

18.14.3 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

18.15 nsec3dig

18.15.1 Synopsis

nsec3dig *IPADDRESS PORT QNAME QTYPE* [recurse]

18.15.2 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.

18.15.3 Example

```
nsec3dig 8.8.8.8 53 doesnotexist.isoc.nl TXT recurse
```

18.16 pdns_control

18.16.1 Synopsis

pdns_control [*OPTION*]... *COMMAND*

18.16.2 Description

pdns_control is used to send commands to a running PowerDNS nameserver.

18.16.3 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.

18.16.4 Commands

bind-add-zone *ZONE FILENAME*

When using the BIND backend, 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-extended-status [*ZONE*. . .]

Output an extended status of all zones, containing much more information than the simple zone status, like the number of records currently loaded, whether pdns is primary or secondary for the zone, the list of primaries, various timers, etc. Optionally, append *ZONE*s to get the status of specific zones.

bind-domain-status [*ZONE*. . .]

When using the BIND backend, list status of all zones. Optionally, append *ZONE*s to get the status of specific zones.

bind-list-rejects

When using the BIND backend, get a list of all rejected zones.

bind-reload-now *ZONE* [*ZONE*. . .]

When using the BIND backend, immediately reload *ZONE* from disk.

ccounts

Show the content of the cache.

current-config [diff]

Show the currently running configuration. The output has the same format as `pdns_server --config`. With the diff option, only modified options are included in the output.

cycle

Restart the nameserver so it reloads its configuration. Only works when the server is running in guardian mode.

list

Dump all statistics and their values in a comma separated list, equivalent to `show *`.

list-zones [primary,secondary,native]

Show a list of zones, optionally filter on the type of zones to show.

notify *ZONE*

Adds *ZONE* to the notification list, causing PowerDNS to send out notifications to the nameservers of a zone. Can be used if a secondary missed previous notifications or is generally hard of hearing. Use * to notify for all zones. (Note that you may need to escape the * sign in your shell.)

notify-host *ZONE 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 output.

quit

Tell a running pdns_server to quit.

rediscover

Instructs backends that new zones 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 *ZONE* [IP]

Retrieve secondary *ZONE* from its primary. Done nearly immediately. If IP is specified, then retrieval is forced from the specified IP. Port may be specified in AFI specific manner.

set *VARIABLE VALUE*

Set the configuration parameter *VARIABLE* to *VALUE*. Currently only the query-logging can be set.

show *STATISTIC*

Show a single statistic, as present in the output of the list command.

show *STATISTIC-PREFIX**

Show all statistics which names start with the supplied *STATISTIC-PREFIX*, as a comma-separated list. Only one (ending) wildcard is allowed.

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.

18.16.5 See also

pdns_server (1), pdnsutil (1)

18.17 pdns_notify

18.17.1 Synopsis

`pdns_notify` *IP_ADDRESS/HOSTNAME*[:*PORT*] *DOMAIN*

18.17.2 Description

`pdns_notify` sends a DNS NOTIFY message to *IP_ADDRESS* or *HOSTNAME*, by default on port 53, for *DOMAIN* and prints the remote nameserver's response. If *HOSTNAME* resolves to multiple IP addresses, each one is notified.

18.17.3 Options

None

18.18 pdns_server

18.18.1 Synopsis

`pdns_server` [*OPTION*]

18.18.2 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.

18.18.3 Options

See the online documentation for all options. The most important ones are:

--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.

--config Show the currently configuration. There are three optional values:

- **--config=default** show the default configuration.
- **--config=diff** show modified options in the current configuration.
- **--config=check** parse the current configuration, with error checking.

--help Display the list of all the available options.

18.18.4 See also

pdns_control(1), pdnsutil(1), https://doc.powerdns.com/

18.19 pdnsutil

18.19.1 Synopsis

pdnsutil [OPTION]… *COMMAND*

18.19.2 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.

18.19.3 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.

18.19.4 Commands

There are many available commands, this section splits them up into their respective uses.

18.19.5 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
- rsasha1-nsec3-sha1
- rsasha256
- rsasha512
- ecdsa256
- ecdsa384
- ed25519
- ed448

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] [published,unpublished] *KEYBITS ALGORITHM*
Create a new key for zone *ZONE*, and make it a KSK or a ZSK (default), with the specified algorithm. The key is inactive by default, set it to **active** to immediately use it to sign *ZONE*. The key is published in the zone by default, set it to **unpublished** to keep it from being returned in a DNSKEY query, which is useful for algorithm rollovers. 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-ds *ZONE* Export to standard output all KSK DS records for *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.

export-zone-key-pem ***ZONE KEY-ID*** Export to standard output full (private) key with key id *KEY-ID* within zone called *ZONE* in the PEM file format. The format is compatible with many non-DNS software products.

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, ECDSA256 is used. If *KEYBITS* is not set, an appropriate keysize is selected for *ALGORITHM*. Each ECC-based algorithm supports only one valid *KEYBITS* value: For ECDSA256 and ED25519, it is 256; for ECDSA384, it is 384; and for ED448, it is 456.

import-zone-key ***ZONE FILE*** **{KSK,ZSK}** Import from *FILE* a full (private) key for the 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.

import-zone-key-pem ***ZONE FILE ALGORITHM*** **{KSK,**ZSK**}** Import from PEM *FILE* a full (private) key for the zone called *ZONE* with a specified *ALGORITHM*. The format used is compatible with many non-DNS software products. **KSK** or **ZSK** specifies the flags this key should have on import. Prints the id of the added key.

publish-zone-key ***ZONE KEY-ID*** Publish the key with id *KEY-ID* within a zone called *ZONE*.

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 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**. And be aware that a high number might overload validating resolvers and that a limit can be set with `max-nsec3-iterations` in `pdns.conf`. 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" (**RFC 7129**) about the next secure record to prevent zone enumeration. Instead of looking it up in the database, it will send out the hash + 1 as the next secure record. Narrow mode requires online signing capabilities by the nameserver and therefore zone transfers are denied. If only the zone is provided as argument, the 4-parameter quoted string defaults to `'1 0 0 -'`. 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.

unpublish-zone-key ***ZONE KEY-ID*** Unpublish the key with id *KEY-ID* within a zone called *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 2 (SHA-256). 0 will publish a CDS with a DNSSEC delete algorithm.

set-publish-cdnskey ***ZONE*** **[delete]** Set *ZONE* to publish CDNSKEY records. Add 'delete' to publish a CDNSKEY with a DNSSEC delete algorithm.

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.

18.19.6 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* {primary,secondary,producer,consumer} Enable TSIG authenticated AXFR using the key *NAME* for zone *ZONE*. This sets the `TSIG-ALLOW-AXFR` (primary/producer) or `AXFR-MASTER-TSIG` (secondary/consumer) zone metadata.

deactivate-tsig-key *ZONE NAME* {primary,secondary,producer,consumer} 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.

18.19.7 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.

add-autoprimary *IP NAMESERVER* [*ACCOUNT*] Add a autoprimary entry into the backend. This enables receiving zone updates from other servers.

remove-autoprimary *IP NAMESERVER* Remove an autoprimary from backend. Not supported by BIND backend.

list-autoprimaries List all autoprimaries.

create-zone *ZONE* Create an empty zone named *ZONE*.

create-secondary-zone *ZONE PRIMARY* [*PRIMARY*]... Create a new secondary zone *ZONE* with primaries *PRIMARY*. All *PRIMARY*s need to to be space-separated IP addresses with an optional port.

change-secondary-zone-primary *ZONE PRIMARY* [*PRIMARY*]... Change the primaries for secondary zone *ZONE* to new primaries *PRIMARY*. All *PRIMARY*s need to to be space-separated 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 zone and settings unchanged

delete-rrset *ZONE NAME TYPE* Delete named RRSET from zone.

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-password [*WORK-FACTOR*] This convenience command asks for a password and returns a hashed and salted version, for use as a webserver password or api key. An optional scrypt work factor can be specified, in power of two, otherwise it defaults to 1024.

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.

increase-serial *ZONE* Increases the SOA-serial by 1. Uses SOA-EDIT.

list-keys [*ZONE*] List DNSSEC information for all keys or for *ZONE*. –verbose or -v will also include the keys for disabled or empty zones.

list-all-zones List all active zone names. –verbose or -v will also include disabled or empty zones.

list-member-zones *CATALOG* List all members of catalog zone *CATALOG*"

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 'ordername' 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 'ordername' 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.

replace-rrset *ZONE NAME TYPE* [*TTL*] *CONTENT* [*CONTENT*. . .] Replace existing *NAME* in zone *ZONE* with a new set.

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* (primary, secondary, native, producer, consumer).

set-options-json *ZONE JSON* Change the options of *ZONE* to *JSON*

set-option *ZONE* [*producer|*consumer*] [*coo**|*unique**|*group*] *VALUE* [*VALUE* . . .]** Set or remove an option for *ZONE*. Providing an empty value removes an option.

set-catalog *ZONE CATALOG* Change the catalog of *ZONE* to *CATALOG*. Setting *CATALOG* to an empty "" removes *ZONE* from the catalog it is in.

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 zonemetadata *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*.

raw-lua-from-content *TYPE CONTENT* Display record contents in a form suitable for dnsdist's *SpoofRawAction*.

zonemd-verify-file *ZONE FILE* Validate ZONEMD for *ZONE* read from *FILE*.

18.19.8 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!

backend-lookup *BACKEND NAME* [*TYPE* [*CLIENT-IP-SUBNET*]] Perform a backend record lookup.

bench-db [*FILE*] Perform a benchmark of the backend-database. *FILE* can be a file with a list, one per line, of zone names to use for this. If *FILE* is not specified, powerdns.com is used.

18.19.9 OTHER TOOLS

b2b-migrate ***OLD NEW*** Migrate data from one backend to another. Needs `launch=OLD,NEW` in the configuration.

ipencrypt ***IP-ADDRESS*** **password** Encrypt an IP address according to the 'ipcipher' standard

ipdecrypt ***IP-ADDRESS*** **password** Decrypt an IP address according to the 'ipcipher' standard

18.19.10 See also

pdns_server (1), pdns_control (1)

18.20 saxfr

18.20.1 Synopsis

`saxfr` *IPADDRESS PORT ZONE* [*Options*]

18.20.2 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.

18.20.3 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.

18.21 sdig

18.21.1 Synopsis

`sdig` *IP-ADDRESS-OR-DOH-URL PORT QNAME QTYPE* [*OPTION*]

18.21.2 Description

`sdig` sends a DNS query to *IP-ADDRESS-OR-DOH-URL* on port *PORT* and displays the answer in a formatted way. If the address starts with an `h`, it is assumed to be a DoH endpoint, and *PORT* is ignored. If qname and qtype are both - and tcp is used, multiple lines are read from stdin, where each line contains a qname and a type. If the address is `stdin`, a DNS packet is read from stdin instead of from the network, and *PORT* is ignored. All input is literal and case-sensitive. Queries need option *recurse* to expect a resource record reply if the query target is not known to be the authoritative server for that record.

18.21.3 Options

These options can be added to the commandline in any order.

class ***CLASSNUM*** Send the query in the numbered class (like 3 for CHAOS) instead of the default 1 (for IN).

dnssec Set the DO bit to request DNSSEC information.

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.

hidesoadetails Don't show the SOA serial in the response.

hidettl Replace TTLs with *[ttl]* in the response.

proxy ***TCP? SRC DST*** Wrap query in PROXYv2 protocol with these parameters. The first parameter accepts 0 for UDP and 1 for TCP. The second and third take IP addresses and port.

recurse Set the RD bit in the question.

showflags Show the NSEC3 flags in the response (they are hidden by default).

dumpluaraw Display record contents in a form suitable for dnsdist's *SpoofRawAction*.

tcp Use TCP instead of UDP to send the query.

dot use DoT instead of UDP to send a query. Implies tcp.

insecure when using DoT, do not validate the server certificate.

fastOpen when using TCP or, DoT, enable TCP Fast Open

subjectName ***name*** when using DoT, verify the server certificate is issued for *name*. The *openssl* provider will accept an empty name and still make sure the certificate is issued by a trusted CA, *gnutls* will only do the validation if a name is given. Default is the empty name. Also, note that older provide libraries might not validate at all.

caStore ***file*** when using DoT, read the trusted CA certificates from *file*. Default is to use the system provided CA store.

tlsProvider ***name*** when using DoT, use TLS provider *name*. Currently supported (if compiled in): *openssl* and *gnutls*. Default is *openssl* if available.

opcode ***OPNUM*** Use opcode *OPNUM* instead of 0 (Query). For example, `sdig 192.0.2.1 53 example.com SOA opcode 4` sends a `NOTIFY`.

18.21.4 Examples

Simple queries to local resolvers `sdig 127.0.0.1 53 example.com AAAA recurse`

`sdig ::1 53 example.com A recurse`

Query to a DNS-over-HTTPS server requesting dnssec and recursion `sdig https://dns.example.net/dns-query 443 example.com A dnssec recurse`

18.22 zone2json

18.22.1 Synopsis

`zone2json` {**–named-conf=***PATH*, **–zone-file=***PATH* [**–zone-name=***NAME*]} [*OPTION*]

18.22.2 Description

zone2json parses BIND named.conf files and zonefiles and outputs JSON on standard output, which can then be fed to the PowerDNS API.

zone2json understands the BIND master file extension `$GENERATE` and will also honour `$ORIGIN` and `$TTL`.

18.22.3 Options

18.22.4 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.

18.22.5 OTHER Options

--help List all options

--on-error-resume-next Ignore missing zone files during parsing. Dangerous.

--verbose Be verbose during conversion.

18.22.6 See also

pdns_server(1)

18.23 zone2ldap

18.23.1 Synopsis

zone2ldap {**–named-conf=***PATH*,**–zone-file=***PATH* **–zone-name=***NAME*} [*OPTION*]. . .

18.23.2 Description

zone2ldap is a program that converts BIND zonefiles to ldif format which can inserted to an LDAP server.

18.23.3 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

18.23.4 See also

pdns_server(1)

18.24 zone2sql

18.24.1 Synopsis

`zone2sql` {**–named-conf=***PATH*,**–zone-file=***PATH* [**–zone-name=***NAME*]} [*Options*]

18.24.2 Description

`zone2sql` parses BIND named.conf files and zonefiles and outputs SQL on standard output, 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 backend.

18.24.3 Options

18.24.4 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.

18.24.5 BACKENDS

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

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

--gsqlite Output in format suitable for the default configuration of the Generic SQLite3 backend.

18.24.6 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.

18.24.7 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.

--secondary 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.

18.24.8 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.

18.24.9 See also

pdns_server(1)

18.25 ixfrdist.yml

18.25.1 Synopsis

ixfrdist.yml

18.25.2 Description

`ixfrdist` reads its configuration from a YAML file. By default, this file is called *ixfrdist.yml* and is read from the directory configured as *SYSCONFDIR* when building the software. This directory is usually one of */etc/pdns*, */etc/powerdns*. Run *ixfrdist –help* to see the default.

18.25.3 Example

```
listen:
  - 192.0.2.2
  - '[2001:DB8:ABCD::2]:5300'
  - 127.0.0.1

acl:
  - 127.0.0.1
  - '192.0.2.0/24'
  - '2001:DB8:ABCD:1234::/64'
```

(continues on next page)

(continued from previous page)

```
work-dir: /var/lib/ixfrdist

uid: ixfrdist
gid: ixfrdist

domains:
  - domain: example.com
    master: 192.0.2.18:5301
    max-soa-refresh: 1800
    notify:
      - 192.0.3.1
      - 192.0.3.2:5301
  - domain: example.net
    master: 2001:DB8:ABCD::2
```

18.25.4 Options

listen The list of addresses to listen on. `ixfrdist` listens on both TCP and UDP. When no port is specified, 53 is used. When specifying ports for IPv6, use the "bracket" notation. By default, `ixfrdist` listens on `127.0.0.1:53` and `[::1]:53`.

acl A list of netmasks that are allowed to query `ixfrdist` and request AXFRs and IXFRs Entries without a netmask will be interpreted as a single address. By default, the ACL is set is `127.0.0.0/8` and `::1/128`.

axfr-max-records Maximum number of records allowed in an AXFR transaction requested by `ixfrdist`. This may prevent untrusted sources from using all the process memory. By default, this setting is `0`, which means "unlimited".

axfr-timeout Timeout in seconds an AXFR transaction requested by `ixfrdist` may take. Increase this when the network to the authoritative servers is slow or the domains are very large and you experience timeouts. Defaults to 20.

failed-soa-retry Time in seconds between retries of the SOA query for a zone we have never transferred. Defaults to 30.

compress Whether record compression should be enabled, leading to smaller answers at the cost of an increased CPU and memory usage. Defaults to false.

work-dir The directory where the domain data is stored. When not set, the current working directory is used. This working directory has the following structure: `work-dir/ZONE/SERIAL`, e.g. `work-dir/rpz.example./2018011902`. It is highly recommended to set this option, as the current working directory might change between invocations. This directory must be writable for the user or group `ixfrdist` runs as.

keep Amount of older copies/IXFR diffs to keep for every domain. This is set to 20 by default.

tcp-in-threads Number of threads to spawn for TCP connections (AXFRs) from downstream hosts. This limits the number of concurrent AXFRs to clients. Set to 10 by default.

gid Group name or numeric ID to drop privileges to after binding the listen sockets. By default, `ixfrdist` runs as the user that started the process.

uid User name or numeric ID to drop privileges to after binding the listen sockets. By default, `ixfrdist` runs as the user that started the process.

domains A list of domains to redistribute. This option is mandatory.

> **domain** The domain name to transfer from the `master`. Mandatory.
>
> **master** IP address of the server to transfer this domain from. Mandatory.
>
> **max-soa-refresh** Cap the refresh time to the given maximum (in seconds). Optional.

notify The list of destinations to send NOTIFY to. Optional.

webserver-address IP address to listen on for the built-in webserver. When not set, no webserver is started.

webserver-acl A list of networks that are allowed to access the **`ixfrdist`** webserver. Entries without a netmask will be interpreted as a single address. By default, this list is set to `127.0.0.0/8` and `::1/128`.

webserver-loglevel How much the webserver should log: 'none', 'normal' or 'detailed'. When logging, each log-line contains the UUID of the request, this allows finding errors caused by certain requests. With 'none', nothing is logged except for errors. With 'normal' (the default), one line per request is logged in the style of the common log format:

```
[NOTICE] [webserver] 46326eef-b3ba-4455-8e76-15ec73879aa3 127.0.0.
↪1:57566 "GET /metrics HTTP/1.1" 200 1846
```

with 'detailed', the full requests and responses (including headers) are logged along with the regular log-line from 'normal'.

18.25.5 See also

`ixfrdist(1)`

CHAPTER

NINETEEN

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`.

19.1 `8bit-dns`

- Boolean
- Default: no

Allow 8 bit DNS queries.

19.2 `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 without TSIG.

> **Warning:** This setting only applies to AXFR without TSIG keys. If you allow a TSIG key to perform an AXFR, this setting will not be checked for that transfer, and the client will be able to perform the AXFR from everywhere.

19.3 `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*.

19.4 `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.

Note: IPs allowed by this setting, still go through the normal NOTIFY processing as described in *Secondary operation* The IP the NOTIFY is received from, still needs to be a nameserver for the secondary domain. Explicitly setting this parameter will not bypass those checks.

19.5 `allow-unsigned-autoprimary`

Changed in version 4.5.0: This was called *allow-unsigned-supermaster* before 4.5.0.

- Boolean
- Default: yes

Turning this off requires all autoprimary notifications to be signed by valid TSIG signature. It will accept any existing key on slave.

19.6 `allow-unsigned-notify`

- Boolean
- Default: yes

Turning this off requires all notifications that are received to be signed by valid TSIG signature for the zone.

19.7 `allow-unsigned-supermaster`

Deprecated since version 4.5.0: Renamed to *allow-unsigned-autoprimary*. Removed in 4.9.0

19.8 `also-notify`

- IP addresses, separated by commas

When notifying a zone, 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*.

You may specify an alternate port by appending :port. Example: `also-notify=192.0.2.1:5300`. If no port is specified, port 53 is used.

19.9 `any-to-tcp`

- Boolean
- Default: yes

Answer questions for the ANY on UDP with a truncated packet that refers the remote server to TCP. Useful for mitigating reflection attacks.

19.10 `api`

- Boolean
- Default: no

Enable/disable the *Built-in Webserver and HTTP API*.

19.11 `api-key`

- String

Changed in version 4.6.0: This setting now accepts a hashed and salted version.

Static pre-shared authentication key for access to the REST API. Since 4.6.0 the key can be hashed and salted using `pdnsutil hash-password` instead of being stored in the configuration in plaintext, but the plaintext version is still supported.

19.12 `autosecondary`

Changed in version 4.5.0: This was called *superslave* before 4.5.0.

- Boolean
- Default: no

Turn on autosecondary support. See *Autoprimary: automatic provisioning of secondaries*.

19.13 `axfr-fetch-timeout`

- Integer
- Default: 10

New in version 4.3.0.

Maximum time in seconds for inbound AXFR to start or be idle after starting.

19.14 `axfr-lower-serial`

- Boolean
- Default: no

Also AXFR a zone from a master with a lower serial.

19.15 `cache-ttl`

- Integer
- Default: 20

Seconds to store packets in the *Packet Cache*. A value of 0 will disable the cache.

19.16 `carbon-instance`

- String
- Default: auth

Set the instance or third string of the metric key. 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*

19.17 `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*.

19.18 `carbon-namespace`

- String
- Default: pdns

Set the namespace or first string of the metric key. 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*

19.19 `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*

19.20 `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). Moreover you can specify more than one server using a comma delimited list, ex: carbon-server=10.10.10.10,10.10.10.20. You may specify an alternate port by appending :port, ex: 127.0.0.1:2004. See *Sending metrics to Graphite/Metronome over Carbon*.

19.21 `chroot`

- Path

If set, chroot to this directory for more security. See *Security of PowerDNS*. This is not recommended; instead, we recommend containing PowerDNS using operating system features. We ship systemd unit files with our packages to make this easy.

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 service to 'simple' instead of 'notify' (refer to the systemd documentation on how to modify unit-files).

19.22 `secondary-check-signature-freshness`

New in version 4.7.0.

- Boolean
- Default: yes

Enabled by default, freshness checks for secondary zones will set the DO flag on SOA queries. PowerDNS can detect (signature) changes on the primary server without serial number bumps using the DNSSEC signatures in the SOA response.

In some problematic scenarios, primary servers send truncated SOA responses. As a workaround, this setting can be turned off, and the DO flag as well as the signature checking will be disabled. To avoid additional drift, primary servers must then always increase the zone serial when it updates signatures.

It is strongly recommended to keep this setting enabled (*yes*).

19.23 `config-dir`

- Path

Location of configuration directory (the directory containing `pdns.conf`). Usually `/etc/powerdns`, but this depends on `SYSCONFDIR` during compile-time.

19.24 `config-name`

- String

Name of this virtual configuration - will rename the binary image. See *Running Virtual Instances*.

19.25 `consistent-backends`

- Boolean
- Default: yes

New in version 4.4.0.

When this is set, PowerDNS assumes that any single zone lives in only one backend. This allows PowerDNS to send `ANY` lookups to its backends, instead of sometimes requesting the exact needed type. This reduces the load on backends by retrieving all the types for a given name at once, adding all of them to the cache. It improves performance significantly for latency-sensitive backends, like SQL ones, where a round-trip takes serious time.

Warning: This behaviour is only a meaningful optimization if the returned response to the `ANY` query can actually be cached, which is not the case if it contains at least one record with a non-zero scope. For this reason `consistent-backends` should be disabled when at least one of the backends in use returns location-based records, like the GeoIP backend.

Note: Pre 4.5.0 the default was no.

19.26 `control-console`

Debugging switch - don't use.

19.27 `daemon`

- Boolean
- Default: no

Operate as a daemon.

19.28 `default-api-rectify`

- Boolean
- Default: yes

The value of *API-RECTIFY* if it is not set on the zone.

Note: Pre 4.2.0 the default was always no.

19.29 `default-catalog-zone`

- String:
- Default: empty

New in version 4.8.3.

When a primary zone is created via the API, and the request does not specify a catalog zone, the name given here will be used.

19.30 `default-ksk-algorithm`

- String
- Default: ecdsa256

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:

- rsasha1
- rsasha256
- rsasha512
- ecdsa256 (ECDSA P-256 with SHA256)
- ecdsa384 (ECDSA P-384 with SHA384)

- ed25519
- ed448

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`.

19.31 `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).

19.32 `default-publish-cdnskey`

- Integer
- Default: empty

New in version 4.3.0.

The default PUBLISH-CDNSKEY value for zones that do not have one individually specified. See the *PUBLISH-CDNSKEY, PUBLISH-CDS* docs for more information.

19.33 `default-publish-cds`

- Comma-separated integers
- Default: empty

New in version 4.3.0.

The default PUBLISH-CDS value for zones that do not have one individually specified. See the *PUBLISH-CDNSKEY, PUBLISH-CDS* docs for more information.

19.34 `default-soa-content`

- String
- Default: a.misconfigured.dns.server.invalid hostmaster.@ 0 10800 3600 604800 3600

New in version 4.4.0.

This value is used when a zone is created without providing a SOA record. @ is replaced by the zone name.

19.35 `default-soa-edit`

- String
- Default: empty

Use this soa-edit value for all zones if no *SOA-EDIT* metadata value is set.

19.36 `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*

19.37 `default-soa-mail`

- String

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Mail address to insert in the SOA record if none set in the backend.

19.38 `default-soa-name`

- String
- Default: a.misconfigured.dns.server.invalid

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Name to insert in the SOA record if none set in the backend.

19.39 `default-ttl`

- Integer
- Default: 3600

TTL to use when none is provided.

19.40 `default-zsk-algorithm`

- String
- Default: (empty)

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:

- rsasha1
- rsasha256
- rsasha512
- ecdsa256 (ECDSA P-256 with SHA256)
- ecdsa384 (ECDSA P-384 with SHA384)
- ed25519
- ed448

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`.

19.41 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).

19.42 delay-notifications

- Integer
- Default: 0 (no delay, send them directly)

Configure a delay to send out notifications, no delay by default.

19.43 direct-dnskey

- Boolean
- Default: no

Read additional DNSKEY, CDS and CDNSKEY records from the records table/your BIND zonefile. If not set, DNSKEY, CDS and CDNSKEY records in the zonefiles are ignored.

19.44 disable-axfr

- Boolean
- Default: no

Do not allow zone transfers.

19.45 disable-axfr-rectify

- Boolean
- Default: no

Disable the rectify step during an outgoing AXFR. Only required for regression testing.

19.46 disable-syslog

- Boolean
- Default: no

Do not log to syslog, only to stderr. 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.

19.47 distributor-threads

- Integer
- Default: 3

Number of Distributor (backend) threads to start per receiver thread. See *Performance and Tuning*.

19.48 dname-processing

- Boolean
- Default: no

Turn on DNAME processing (DNAME substitution, CNAME synthesis). This approximately doubles query load.

If this is turned off, DNAME records are treated as any other and served only when queried explicitly.

19.49 dnsproxy-udp-port-range

- String
- Default: *10000 60000*

If *resolver* enables the DNS Proxy, this setting limits the port range the DNS Proxy's UDP port is chosen from.

Default should be fine on most installs, but if you have conflicting local services, you may choose to limit the range.

19.50 dnssec-key-cache-ttl

- Integer
- Default: 30

Seconds to cache DNSSEC keys from the database. A value of 0 disables caching.

19.51 dnsupdate

- Boolean
- Default: no

Enable/Disable DNS update (RFC2136) support. See *Dynamic DNS Update (RFC 2136)* for more.

19.52 dnsupdate-require-tsig

New in version 5.0.0.

- Boolean
- Default: no

Requires DNS updates to be signed by a valid TSIG signature even if the zone has no associated keys.

19.53 do-ipv6-additional-processing

- Boolean
- Default: yes

Changed in version 4.4.0: This setting has been removed

Perform AAAA additional processing. This sends AAAA records in the ADDITIONAL section when sending a referral.

19.54 domain-metadata-cache-ttl

Deprecated since version 4.5.0: Renamed to *zone-metadata-cache-ttl*.

Seconds to cache zone metadata from the database. A value of 0 disables caching.

19.55 edns-cookie-secret

New in version 4.6.0.

- String
- Default: (empty)

When set, PowerDNS will respond with **RFC 9018** EDNS Cookies to queries that have the EDNS0 Cookie option. PowerDNS will also respond with BADCOOKIE to clients that have sent only a client cookie, or a bad server cookie (section 5.2.3 and 5.2.4 of **RFC 7873**).

This setting MUST be 32 hexadecimal characters, as the siphash algorithm's key used to create the cookie requires a 128-bit key.

19.56 edns-subnet-processing

- Boolean
- Default: no

Enables EDNS subnet processing, for backends that support it.

19.57 enable-gss-tsig

- Boolean
- Default: no

Enable accepting GSS-TSIG signed messages. In addition to this setting, see *TSIG*.

19.58 enable-lua-records

- One of `no`, `yes` (or empty), or `shared`, String
- Default: no

Globally enable the *LUA records* feature.

To use shared LUA states, set this to `shared`, see *Shared Lua state model*.

19.59 entropy-source

- Path
- Default: /dev/urandom

Entropy source file to use.

19.60 expand-alias

- Boolean
- Default: no

If this is enabled, ALIAS records are expanded (synthesized to their A/AAAA).

If this is disabled (the default), ALIAS records will not be 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.

19.61 resolve-across-zones

New in version 5.0.0.

- Boolean
- Default: yes

If this is enabled, CNAME records and other referrals will be resolved as long as their targets exist in any local backend. Can be disabled to allow for different authorities managing zones in the same server instance.

Referrals not available in local backends are never resolved. SVCB referrals are never resolved across zones. ALIAS is not impacted by this setting.

19.62 forward-dnsupdate

- Boolean
- Default: no

Forward DNS updates sent to a slave to the master.

19.63 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.

19.64 `guardian`

- Boolean
- Default: no

Run within a guardian process. See *Guardian*.

19.65 `ignore-unknown-settings`

New in version 4.5.0.

- Setting names, separated by commas
- Default: empty

Names of settings to be ignored while parsing configuration files, if the setting name is unknown to PowerDNS.

Useful during upgrade testing.

19.66 `include-dir`

- Path

Directory to scan for additional config files. All files that end with .conf are loaded in order using `POSIX` as locale.

19.67 `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.

Running multiple instances of the BIND backend is not allowed.

19.68 `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.

19.69 `local-address`

Changed in version 4.3.0: now also accepts IPv6 addresses

Changed in version 4.3.0: Before 4.3.0, this setting only supported IPv4 addresses.

- IPv4/IPv6 Addresses, with optional port numbers, separated by commas or whitespace
- Default: `0.0.0.0, ::`

Local IP addresses to which we bind. Each address specified can include a port number; if no port is included then the *local-port* port will be used for that address. If a port number is specified, it must be separated from the address with a ':'; for an IPv6 address the address must be enclosed in square brackets.

Examples:

```
local-address=127.0.0.1 ::1
local-address=0.0.0.0:5353
local-address=[::]:8053
local-address=127.0.0.1:53, [::1]:5353
```

19.70 `local-address-nonexist-fail`

- Boolean
- Default: yes

Fail to start if one or more of the *local-address*'s do not exist on this server.

19.71 `local-ipv6`

Deprecated since version 4.5.0: Use *local-address* instead

19.72 `local-ipv6-nonexist-fail`

Changed in version 4.3.0: This setting has been removed, use *local-address-nonexist-fail*

- Boolean
- Default: no

Fail to start if one or more of the *local-ipv6* addresses do not exist on this server.

19.73 `local-port`

- Integer
- Default: 53

Local port to bind to. If an address in *local-address* does not have an explicit port, this port is used.

19.74 `log-dns-details`

- Boolean
- Default: no

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

19.75 `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.

19.76 `log-timestamp`

- Bool
- Default: yes

When printing log lines to stderr, 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

19.77 `logging-facility`

If set to a digit, logging is performed under this LOCAL facility. See *Logging to syslog*. Do not pass names like 'local0'!

19.78 `loglevel`

- Integer
- Default: 4

Amount of logging. The higher the number, the more lines logged. Corresponds to "syslog" level values (e.g. 0 = emergency, 1 = alert, 2 = critical, 3 = error, 4 = warning, 5 = notice, 6 = info, 7 = debug). Each level includes itself plus the lower levels before it. Not recommended to set this below 3.

19.79 `loglevel-show`

- Bool
- Default: no

New in version 4.9.0.

When enabled, log messages are formatted like structured logs, including their log level/priority: `msg="Unable to launch, no backends configured for querying" prio="Error"`

19.80 `lua-axfr-script`

- String
- Default: empty

Script to be used to edit incoming AXFRs, see *Modifying a secondary zone using a script*

19.81 `lua-consistent-hashes-cleanup-interval`

- Integer
- Default: 3600

New in version 4.9.0.

Amount of time (in seconds) between subsequent cleanup routines for pre-computed hashes related to *`pickchashed()`*.

19.82 `lua-consistent-hashes-expire-delay`

- Integer
- Default: 86400

New in version 4.9.0.

Amount of time (in seconds) a pre-computed hash entry will be considered as expired when unused. See *`pickchashed()`*.

19.83 `lua-global-include-dir`

- String
- Default: empty
- Example: `/etc/pdns/lua-global/`

New in version 5.0.0.

When creating a Lua context, scan this directory for additional lua files. All files that end with .lua are loaded in order using `POSIX` as locale with Lua scripts.

19.84 `lua-health-checks-expire-delay`

- Integer
- Default: 3600

New in version 4.3.0.

Amount of time (in seconds) to expire (remove) a LUA monitoring check when the record isn't used any more (either deleted or modified).

19.85 `lua-health-checks-interval`

- Integer
- Default: 5

New in version 4.3.0.

Amount of time (in seconds) between subsequent monitoring health checks. Does nothing if the checks take more than that time to execute.

19.86 `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.

19.87 `lua-records-exec-limit`

- Integer
- Default: 1000

Limit LUA records scripts to `lua-records-exec-limit` instructions. Setting this to any value less than or equal to 0 will set no limit.

19.88 `lua-records-insert-whitespace`

- Boolean
- Default: no in 5.0, yes before that

New in version 4.9.1.

When combining the `"` delimited chunks of a LUA record, whether to insert whitespace between each chunk.

19.89 `master`

Deprecated since version 4.5.0: Renamed to *primary*. Removed in 4.9.0.

- Boolean
- Default: no

Turn on master support. See *Primary operation*.

19.90 `max-cache-entries`

- Integer
- Default: 1000000

Maximum number of entries in the query cache. 1 million (the default) will generally suffice for most installations.

19.91 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.

19.92 max-include-depth

- Integer
- Default: 20

Maximum number of nested `$INCLUDE` directives while processing a zone file. Zero mean no `$INCLUDE` directives will be accepted.

19.93 max-generate-steps

- Integer
- Default: 0

Maximum number of steps for a '$GENERATE' directive when parsing a zone file. This is a protection measure to prevent consuming a lot of CPU and memory when untrusted zones are loaded. Default to 0 which means unlimited.

19.94 max-nsec3-iterations

- Integer
- Default: 100

Limit the number of NSEC3 hash iterations for zone configurations. For more information see *Setting the NSEC modes and parameters*.

Note: Pre 4.5.0 the default was 500.

19.95 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.

19.96 max-queue-length

- Integer
- Default: 5000

If this many packets are waiting for database attention, consider the situation hopeless and respawn the server process. This limit is per receiver thread.

19.97 `max-signature-cache-entries`

- Integer
- Default: 2^31-1 (on most systems), 2^63-1 (on ILP64 systems)

Maximum number of DNSSEC signature cache entries. This cache is automatically reset once per week or when the cache is full. If you use NSEC narrow mode, this cache can grow large.

19.98 `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.

19.99 `max-tcp-connections`

- Integer
- Default: 20

Allow this many incoming TCP DNS connections simultaneously.

19.100 `max-tcp-connections-per-client`

- Integer
- Default: 0

Maximum number of simultaneous TCP connections per client. 0 means unlimited.

19.101 `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.

19.102 `module-dir`

- Path

Directory for modules. Default depends on `PKGLIBDIR` during compile-time.

19.103 `negquery-cache-ttl`

- Integer
- Default: 60

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

19.104 `no-config`

- Boolean
- Default: no

Do not attempt to read the configuration file. Useful for configuration by parameters from the command line only.

19.105 `no-shuffle`

- Boolean
- Default: no

Do not attempt to shuffle query results, used for regression testing.

19.106 `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.

19.107 `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-renotify 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` zone metadata always receive AXFR NOTIFYs.

IP addresses and netmasks can be excluded by prefixing them with a `!`. To notify all IP addresses apart from the 192.168.0.0/24 subnet use the following:

```
only-notify=0.0.0.0/0, ::/0, !192.168.0.0/24
```

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* zone metadata to avoid this potential bottleneck.

Note: If your slaves support an Internet Protocol version, which your master does not, then set `only-notify` to include only supported protocol version. Otherwise there will be error trying to resolve address.

For example, slaves support both IPv4 and IPv6, but PowerDNS master have only IPv4, so allow only IPv4 with `only-notify`:

```
only-notify=0.0.0.0/0
```

19.108 `outgoing-axfr-expand-alias`

- One of `no`, `yes`, or `ignore-errors`, String
- Default: no

Changed in version 4.9.0: Option *ignore-errors* added.

If this is enabled, ALIAS records are expanded (synthesized 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.

If the ALIAS target cannot be resolved during AXFR the AXFR will fail. To allow outgoing AXFR also if the ALIAS targets are broken set this setting to *ignore-errors*. Be warned, this will lead to inconsistent zones between Primary and Secondary name servers.

19.109 `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. Packets not in the cache will be dropped, and *overload-drops* will be incremented.

19.110 `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”.

19.111 `primary`

Changed in version 4.5.0: This was called *master* before 4.5.0.

- Boolean
- Default: no

Turn on operating as a primary. See *Primary operation*.

19.112 `proxy-protocol-from`

New in version 4.6.0.

- IP addresses or netmasks, separated by commas
- Default: empty

Ranges that are required to send a Proxy Protocol version 2 header in front of UDP and TCP queries, to pass the original source and destination addresses and ports to the Authoritative. Queries that are not prefixed with such a header will not be accepted from clients in these ranges. Queries prefixed by headers from clients that are not listed in these ranges will be dropped.

Note that once a Proxy Protocol header has been received, the source address from the proxy header instead of the address of the proxy will be checked against primary addresses sending NOTIFYs, and the ACLs for any client requesting AXFRs. When using this setting combined with *trusted-notification-proxy*, please be aware that the trusted address will also be checked against the source address in the PROXY header.

The dnsdist docs have more information about the PROXY protocol.

19.113 `proxy-protocol-maximum-size`

New in version 4.6.0.

- Integer
- Default: 512

The maximum size, in bytes, of a Proxy Protocol payload (header, addresses and ports, and TLV values). Queries with a larger payload will be dropped.

19.114 `query-cache-ttl`

- Integer
- Default: 20

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

19.115 `query-local-address`

Changed in version 4.4.0: Accepts both IPv4 and IPv6 addresses. Also accept more than one address per address family.

- IP addresses, separated by spaces or commas
- Default: *0.0.0.0 ::*

The IP addresses 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.

PowerDNS will pick the correct address family based on the remote's address (v4 for outgoing v4, v6 for outgoing v6). However, addresses are selected at random without taking into account ip subnet reachability. It is highly recommended to use the defaults in that case (the kernel will pick the right source address for the network).

19.116 `query-local-address6`

Deprecated since version 4.5.0: Removed. Use *query-local-address*.

19.117 `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.

19.118 `queue-limit`

- Integer
- Default: 1500

Maximum number of milliseconds to queue a query. See *Performance and Tuning*.

19.119 `receiver-threads`

- Integer
- Default: 1

Number of receiver (listening) threads to start. See *Performance and Tuning*.

19.120 `resolver`

- IP Address with optional port
- Default: unset

Recursive DNS server to use for ALIAS lookups and the internal stub resolver. Only one address can be given.

It is assumed that the specified recursive DNS server, and the network path to it, are trusted.

Examples:

```
resolver=127.0.0.1
resolver=[::1]:5300
```

19.121 `retrieval-threads`

- Integer
- Default: 2

Number of AXFR slave threads to start.

19.122 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.

19.123 rng

- String
- Default: auto

Specify which random number generator to use. Permissible choices are:

- auto - choose automatically
- sodium - Use libsodium `randombytes_uniform`
- openssl - Use libcrypto `RAND_bytes`
- getrandom - Use libc getrandom, falls back to urandom if it does not really work
- arc4random - Use BSD `arc4random_uniform`
- urandom - Use `/dev/urandom`
- kiss - Use simple settable deterministic RNG. **FOR TESTING PURPOSES ONLY!**

Note: Not all choices are available on all systems.

19.124 secondary

Changed in version 4.5.0: This was called *slave* before 4.5.0.

- Boolean
- Default: no

Turn on operating as a secondary. See *Secondary operation*.

19.125 secondary-do-renotify

Changed in version 4.5.0: This was called *slave-renotify* before 4.5.0.

- Boolean
- Default: no

This setting will make PowerDNS renotify the secondaries after an AXFR is *received* from a primary. This is useful, among other situations, when running a signing secondary.

See *SLAVE-RENOTIFY* to set this per-zone.

19.126 security-poll-suffix

- String
- Default: secpoll.powerdns.com.

Zone name from which to query security update notifications. Setting this to an empty string disables secpoll.

19.127 send-signed-notify

- Boolean
- Default: yes

If yes, outgoing NOTIFYs will be signed if a TSIG key is configured for the zone. If there are multiple TSIG keys configured for a zone, PowerDNS will use the first one retrieved from the backend, which may not be the correct one for the respective slave. Hence, in setups with multiple slaves with different TSIG keys it may be required to send NOTIFYs unsigned.

19.128 server-id

- String
- Default: The hostname of the server

This is the server ID that will be returned on an EDNS NSID query.

19.129 setgid

- String

If set, change group id to this gid for more security. See *Security of PowerDNS*.

19.130 setuid

- String

If set, change user id to this uid for more security. See *Security of PowerDNS*.

19.131 signing-threads

- Integer
- Default: 3

Tell PowerDNS how many threads to use for signing. It might help improve signing speed by changing this number.

19.132 slave

Deprecated since version 4.5.0: Renamed to *secondary*. Removed in 4.9.0.

19.133 `slave-cycle-interval`

Deprecated since version 4.5.0: Renamed to *xfr-cycle-interval*. Removed in 4.9.0.

19.134 `slave-renotify`

Deprecated since version 4.5.0: Renamed to *secondary-do-renotify*. Removed in 4.9.0.

- Boolean
- Default: no

This setting will make PowerDNS renotify the slaves after an AXFR is *received* from a master. This is useful when running a signing-slave.

See *SLAVE-RENOTIFY* to set this per-zone.

19.135 `soa-expire-default`

- Integer
- Default: 604800

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Default *SOA* expire.

19.136 `soa-minimum-ttl`

- Integer
- Default: 3600

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Default *SOA* minimum ttl.

19.137 `soa-refresh-default`

- Integer
- Default: 10800

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Default *SOA* refresh.

19.138 `soa-retry-default`

- Integer
- Default: 3600

Deprecated since version 4.2.0: This setting has been removed in 4.4.0

Default *SOA* retry.

19.139 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.

19.140 superslave

Deprecated since version 4.5.0: Renamed to *autosecondary*. Removed in 4.9.0.

- Boolean
- Default: no

Turn on supermaster support. See *Autoprimary: automatic provisioning of secondaries*.

19.141 svc-autohints

- Boolean
- Default: no

New in version 4.5.0.

Whether or not to enable IPv4 and IPv6 *autohints*.

19.142 tcp-control-address

- IP Address

Address to bind to for TCP control.

19.143 tcp-control-port

- Integer
- Default: 53000

Port to bind to for TCP control.

19.144 tcp-control-range

- IP Ranges, separated by commas or whitespace

Limit TCP control to a specific client range.

19.145 `tcp-control-secret`

- String

Password for TCP control.

19.146 `tcp-fast-open`

- Integer
- Default: 0 (Disabled)

Enable TCP Fast Open support, if available, on the listening sockets. The numerical value supplied is used as the queue size, 0 meaning disabled.

19.147 `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.

19.148 `traceback-handler`

- Boolean
- Default: yes

Enable the Linux-only traceback handler.

19.149 `trusted-notification-proxy`

Changed in version 4.4.0: This option now accepts a comma-separated list of IP ranges. This was a list of IP addresses before.

- IP ranges, separated by commas

IP ranges of incoming notification proxies.

19.150 `udp-truncation-threshold`

- Integer
- Default: 1232

EDNS0 allows for large UDP response datagrams, which can potentially raise performance. Large responses however also have downsides in terms of reflection attacks. Maximum value is 65535, but values above 4096 should probably not be attempted.

Note: Why 1232?

1232 is the largest number of payload bytes that can fit in the smallest IPv6 packet. IPv6 has a minimum MTU of 1280 bytes (**RFC 8200, section 5**), minus 40 bytes for the IPv6 header, minus 8 bytes for the UDP header gives 1232, the maximum payload size for the DNS response.

19.151 `upgrade-unknown-types`

- Boolean
- Default: no

New in version 4.4.0.

Transparently upgrade records stored as *TYPE#xxx* and RFC 3597 (hex format) contents, if the type is natively supported. When this is disabled, records stored in this format cannot be served.

Recommendation: keep disabled for better performance. Enable for testing PowerDNS upgrades, without changing stored records. Enable for upgrading record content on secondaries, or when using the API (see *upgrade notes*). Disable after record contents have been upgraded.

This option is supported by the bind and Generic SQL backends.

Note: When using a generic SQL backend, records with an unknown record type (see *Supported Record Types*) can be identified with the following SQL query:

```
SELECT * from records where type like 'TYPE%';
```

19.152 `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 - https://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.

19.153 `webserver`

- Boolean
- Default: no

Start a webserver for monitoring. See *Performance and Tuning*".

19.154 `webserver-address`

- IP Address
- Default: 127.0.0.1

IP Address for webserver/API to listen on.

19.155 webserver-allow-from

- IP ranges, separated by commas or whitespace
- Default: 127.0.0.1,::1

Webserver/API access is only allowed from these subnets.

19.156 webserver-hash-plaintext-credentials

New in version 4.6.0.

- Boolean
- Default: no

Whether passwords and API keys supplied in the configuration as plaintext should be hashed during startup, to prevent the plaintext versions from staying in memory. Doing so increases significantly the cost of verifying credentials and is thus disabled by default. Note that this option only applies to credentials stored in the configuration as plaintext, but hashed credentials are supported without enabling this option.

19.157 webserver-loglevel

- String, one of “none”, “normal”, “detailed”
- Default: normal

The amount of logging the webserver must do. “none” means no useful webserver information will be logged. When set to “normal”, the webserver will log a line per request that should be familiar:

```
[webserver] e235780e-a5cf-415e-9326-9d33383e739e 127.0.0.1:55376 "GET /api/v1/
↪servers/localhost/bla HTTP/1.1" 404 196
```

When set to “detailed”, all information about the request and response are logged:

```
[webserver] e235780e-a5cf-415e-9326-9d33383e739e Request Details:
[webserver] e235780e-a5cf-415e-9326-9d33383e739e  Headers:
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   accept: text/html,application/
↪xhtml+xml,application/xml;q=0.9,*/*;q=0.8
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   accept-encoding: gzip, deflate
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   accept-language: en-US,en;q=0.5
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   connection: keep-alive
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   dnt: 1
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   host: 127.0.0.1:8081
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   upgrade-insecure-requests: 1
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   user-agent: Mozilla/5.0 (X11;␣
↪Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
[webserver] e235780e-a5cf-415e-9326-9d33383e739e  No body
[webserver] e235780e-a5cf-415e-9326-9d33383e739e Response details:
[webserver] e235780e-a5cf-415e-9326-9d33383e739e  Headers:
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   Connection: close
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   Content-Length: 49
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   Content-Type: text/html;␣
↪charset=utf-8
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   Server: PowerDNS/0.0.15896.0.
↪gaba8bab3ab
[webserver] e235780e-a5cf-415e-9326-9d33383e739e  Full body:
[webserver] e235780e-a5cf-415e-9326-9d33383e739e   <!html><title>Not Found</title>
↪<h1>Not Found</h1>
[webserver] e235780e-a5cf-415e-9326-9d33383e739e 127.0.0.1:55376 "GET /api/v1/
↪servers/localhost/bla HTTP/1.1" 404 196
```

(continues on next page)

(continued from previous page)

The value between the hooks is a UUID that is generated for each request. This can be used to find all lines related to a single request.

Note: The webserver logs these line on the NOTICE level. The *loglevel* setting must be 5 or higher for these lines to end up in the log.

19.158 `webserver-max-bodysize`

- Integer
- Default: 2

Maximum request/response body size in megabytes.

19.159 `webserver-connection-timeout`

New in version 4.8.5.

- Integer
- Default: 5

Request/response timeout in seconds.

19.160 `webserver-password`

Changed in version 4.6.0: This setting now accepts a hashed and salted version.

- String

Password required to access the webserver. Since 4.6.0 the password can be hashed and salted using `pdnsutil hash-password` instead of being present in the configuration in plaintext, but the plaintext version is still supported.

19.161 `webserver-port`

- Integer
- Default: 8081

The port where webserver/API will listen on.

19.162 `webserver-print-arguments`

- Boolean
- Default: no

If the webserver should print arguments.

19.163 `write-pid`

- Boolean
- Default: yes

If a PID file should be written.

19.164 `workaround-11804`

- Boolean
- Default: no

Workaround for issue #11804 (outgoing AXFR may try to overfill a chunk and fail).

Default of no implies the pre-4.8 behaviour of up to 100 RRs per AXFR chunk.

If enabled, only a single RR will be put into each AXFR chunk, making some zones transferable when they were not otherwise.

19.165 `xfr-cycle-interval`

Changed in version 4.5.0: This was called *slave-cycle-interval* before 4.5.0.

- Integer
- Default: 60

On a primary, this is the amount of seconds between the primary checking the SOA serials in its database to determine to send out NOTIFYs to the secondaries. On secondaries, this is the number of seconds between the secondary checking for updates to zones.

19.166 `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.

19.167 `zone-cache-refresh-interval`

- Integer
- Default: 300

Seconds to cache a list of all known zones. A value of 0 will disable the cache.

If your backends do not respond to unknown or dynamically generated zones, it is suggested to enable *consistent-backends* (default since 4.5) and leave this option at its default of *300*.

19.168 `zone-metadata-cache-ttl`

Changed in version 4.5.0: This was called *domain-metadata-cache-ttl* before 4.5.0.

- Integer
- Default: 60

Seconds to cache zone metadata from the database. A value of 0 disables caching.

CHAPTER

TWENTY

SECURITY ADVISORIES

All security advisories for the PowerDNS Authoritative Server are listed here.

20.1 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.

20.2 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.

20.3 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.

20.4 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`` <../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.

20.5 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.

20.6 PowerDNS Security Advisory 2015-03: Packet parsing bug can lead to crashes

- CVE: CVE-2015-5311
- Date: November 9th 2015
- Credit: Chris 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 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 Chris Hofstaedtler of Deduktiva GmbH for finding and reporting this issue.

20.7 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.

20.8 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 dnsdist 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.

dnsdist 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.

20.9 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.

20.10 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.

20.11 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.

20.12 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.

20.13 PowerDNS Security Advisory 2018-02: Buffer overflow in dnsreplay

- CVE: CVE-2018-1046
- Date: May 8th 2018
- Credit: Wei Hao

- Affects: dnsreplay from 4.0.0 up to and including 4.1.1
- Not affected: dnsreplay 3.4.11, 4.1.2
- Severity: High
- Impact: Arbitrary code execution
- Exploit: This problem can be triggered via a crafted PCAP file
- Risk of system compromise: Yes
- Solution: Upgrade to a non-affected version

An issue has been found in the dnsreplay tool provided with PowerDNS Authoritative, where replaying a specially crafted PCAP file can trigger a stack-based buffer overflow, leading to a crash and potentially arbitrary code execution. This buffer overflow only occurs when the *–ecs-stamp* option of dnsreplay is used. Regardless of this issue, the use of dnsreplay with untrusted PCAP files is not advised. This issue has been assigned CVE-2018-1046 by Red Hat.

PowerDNS Authoritative from 4.0.0 up to and including 4.1.1 is affected.

We would like to thank Wei Hao for finding and subsequently reporting this issue.

20.14 PowerDNS Security Advisory 2018-03: Crafted zone record can cause a denial of service

- CVE: CVE-2018-10851
- Date: November 6th 2018
- Affects: PowerDNS Authoritative from 3.3.0 up to and including 4.1.4
- Not affected: 4.1.5, 4.0.6
- Severity: Medium
- Impact: Denial of service
- Exploit: This problem can be triggered via crafted records
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: run the process inside the guardian or inside a supervisor

An issue has been found in PowerDNS Authoritative Server allowing an authorized user to cause a memory leak 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 the fact that some memory is allocated before the parsing and is not always properly released if the record is malformed.

This issue has been assigned CVE-2018-10851.

When the PowerDNS Authoritative Server is run inside the guardian (`--guardian`), or inside a supervisor like supervisord or systemd, an out-of-memory crash will lead to an automatic restart, limiting the impact to a somewhat degraded service.

PowerDNS Authoritative from 3.3.0 up to and including 4.1.4 is affected. Please note that at the time of writing, PowerDNS Authoritative 3.4 and below are no longer supported, as described in *End of life statements*.

20.15 PowerDNS Security Advisory 2018-05: Packet cache pollution via crafted query

- CVE: CVE-2018-14626

- Date: November 6th 2018
- Affects: PowerDNS Authoritative from 4.1.0 up to and including 4.1.4
- Not affected: 4.1.5, 4.0.x
- Severity: Medium
- Impact: Denial of service
- Exploit: This problem can be triggered via crafted queries
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in PowerDNS Authoritative Server allowing a remote user to craft a DNS query that will cause an answer without DNSSEC records to be inserted into the packet cache and be returned to clients asking for DNSSEC records, thus hiding the presence of DNSSEC signatures for a specific qname and qtype. For a DNSSEC-signed domain, this means that DNSSEC validating clients will consider the answer to be bogus until it expires from the packet cache, leading to a denial of service.

This issue has been assigned CVE-2018-14626.

PowerDNS Authoritative from 4.1.0 up to and including 4.1.4 is affected.

We would like to thank Kees Monshouwer for finding and subsequently reporting this issue.

20.16 PowerDNS Security Advisory 2019-03: Insufficient validation in the HTTP remote backend

- CVE: CVE-2019-3871
- Date: March 18th 2019
- Affects: PowerDNS Authoritative up to and including 4.1.6
- Not affected: 4.1.7, 4.0.7
- Severity: High
- Impact: Denial of Service, Information Disclosure, Content spoofing
- Exploit: This problem can be triggered via crafted queries
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in PowerDNS Authoritative Server when the HTTP remote backend is used in RESTful mode (without post=1 set), allowing a remote user to cause the HTTP backend to connect to an attacker-specified host instead of the configured one, via a crafted DNS query. This can be used to cause a denial of service by preventing the remote backend from getting a response, content spoofing if the attacker can time its own query so that subsequent queries will use an attacker-controlled HTTP server instead of the configured one, and possibly information disclosure if the Authoritative Server has access to internal servers.

This issue has been assigned CVE-2019-3871.

PowerDNS Authoritative up to and including 4.1.6 is affected. Please note that at the time of writing, PowerDNS Authoritative 3.4 and below are no longer supported, as described in https://doc.powerdns.com/authoritative/appendices/EOL.html.

We would like to thank Adam Dobrawy, Frederico Silva and Gregory Brzeski from HyperOne.com for finding and subsequently reporting this issue!

20.17 PowerDNS Security Advisory 2019-04: Denial of service via crafted zone records

- CVE: CVE-2019-10162
- Date: June 21st 2019
- Affects: PowerDNS Authoritative up to and including 4.1.9
- Not affected: 4.1.10, 4.0.8
- Severity: Medium
- Impact: Denial of Service
- Exploit: This problem can be triggered via crafted records
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: run the process inside the guardian or inside a supervisor

An issue has been found in PowerDNS Authoritative Server allowing an authorized user to cause the server to exit by inserting a crafted record in a MASTER type zone under their control. The issue is due to the fact that the Authoritative Server will exit when it runs into a parsing error while looking up the NS/A/AAAA records it is about to use for an outgoing notify.

This issue has been assigned CVE-2019-10162.

PowerDNS Authoritative up to and including 4.1.9 is affected. Please note that at the time of writing, PowerDNS Authoritative 3.4 and below are no longer supported, as described in https://doc.powerdns.com/authoritative/appendices/EOL.html.

We would like to thank Gert van Dijk for finding and subsequently reporting this issue!

20.18 PowerDNS Security Advisory 2019-05: Denial of service via NOTIFY packets

- CVE: CVE-2019-10163
- Date: June 21st 2019
- Affects: PowerDNS Authoritative up to and including 4.1.8
- Not affected: 4.1.9, 4.0.8
- Severity: Medium
- Impact: Denial of Service
- Exploit: This problem can be triggered via the sending of NOTIFY packets from an authorized master
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in PowerDNS Authoritative Server allowing a remote, authorized master server to cause a high CPU load or even prevent any further updates to any slave zone by sending a large number of NOTIFY messages. Note that only servers configured as slaves are affected by this issue.

This issue has been assigned CVE-2019-10163.

PowerDNS Authoritative up to and including 4.1.8 is affected. Please note that at the time of writing, PowerDNS Authoritative 3.4 and below are no longer supported, as described in https://doc.powerdns.com/authoritative/appendices/EOL.html.

We would like to thank George Asenov for finding and subsequently reporting this issue!

20.19 PowerDNS Security Advisory 2019-06: Denial of service via crafted zone records

- CVE: CVE-2019-10203
- Date: July 30th, 2019
- Affects: PowerDNS Authoritative 4.0.0 and up, when using the gpgsql (PostgreSQL) backend
- Not affected: 4.2.0, 4.1.11, 4.0.9
- Severity: Low
- Impact: Denial of Service
- Exploit: This problem can be triggered via crafted records
- Risk of system compromise: No
- Solution: Update the database schema
- Workaround: run the process inside the guardian or inside a supervisor

An issue has been found in PowerDNS Authoritative Server allowing an authorized user to cause the server to exit by inserting a crafted record in a MASTER type zone under their control. The issue is due to the fact that the Authoritative Server will exit when it tries to store the notified serial in the PostgreSQL database, if this serial cannot be represented in 31 bits.

This issue has been assigned CVE-2019-10203.

PowerDNS Authoritative up to and including 4.1.10 is affected. Please note that at the time of writing, PowerDNS Authoritative 3.4 and below are no longer supported, as described in https://doc.powerdns.com/authoritative/appendices/EOL.html.

To fix the issue, run the following command against your PostgreSQL pdns database: *ALTER TABLE domains ALTER notified_serial TYPE bigint USING CASE WHEN notified_serial >= 0 THEN notified_serial::bigint END;*. No software changes are required.

We would like to thank Klaus Darilion for finding and subsequently reporting this issue!

20.20 PowerDNS Security Advisory 2020-05: Leaking uninitialised memory through crafted zone records

- CVE: CVE-2020-17482
- Date: September 22nd, 2020
- Affects: PowerDNS Authoritative 4.3.0 and earlier
- Not affected: 4.3.1 and up, 4.2.3 and up, 4.1.14 and up
- Severity: Low
- Impact: Information leak
- Exploit: This problem can be triggered via crafted records by an authorized user
- Risk of system compromise: Low
- Solution: Upgrade to a fixed version
- Workaround: Do not take zone data from untrusted users

An issue has been found in PowerDNS Authoritative Server before 4.3.1 where an authorized user with the ability to insert crafted records into a zone might be able to leak the content of uninitialized memory. Such a user could be a customer inserting data via a control panel, or somebody with access to the REST API. Crafted records cannot be inserted via AXFR.

This issue has been assigned CVE-2020-17482.

PowerDNS Authoritative up to and including version 4.3.0 are affected. Please note that at the time of writing, PowerDNS Authoritative 4.0 and below are no longer supported, as described in https://doc.powerdns.com/authoritative/appendices/EOL.html.

We would like to thank Nathaniel Ferguson for finding and subsequently reporting this issue!

20.21 PowerDNS Security Advisory 2020-06: Various issues in our GSS-TSIG support

- CVE: CVE-2020-24696, CVE-2020-24697, CVE-2020-24698
- Date: September 22nd, 2020
- Affects: PowerDNS Authoritative versions before 4.4.0, when compiled with –enable-experimental-gss-tsig
- Not affected: 4.4.0 and up, and any version compiled without GSS-TSIG support
- Severity: Low
- Impact: Crashes, Information Leaks, Possible code execution
- Exploit: This problem can be triggered via crafted packets
- Risk of system compromise: Low
- Solution: Do not use software built with GSS-TSIG support

Various issues have been found in our GSS-TSIG support, where an unauthorized attacker could cause crashes, possibly leak uninitialised memory, and possibly execute arbitrary code.

These issues have been assigned:

- CVE-2020-24696: A remote, unauthenticated attacker can trigger a race condition leading to a crash, or possibly arbitrary code execution, by sending crafted queries with a GSS-TSIG signature.
- CVE-2020-24697: A remote, unauthenticated attacker can cause a denial of service by sending crafted queries with a GSS-TSIG signature.
- CVE-2020-24698: A remote, unauthenticated attacker might be able to cause a double-free, leading to a crash or possibly arbitrary code execution by sending crafted queries with a GSS-TSIG signature.

All PowerDNS Authoritative versions are affected, but *only* if they have been compiled with `--enable-experimental-gss-tsig`. We have never published packages with the feature enabled.

Because of the various issues with the feature (including a complete lack of testing code around it), and no reports of production usage of GSS-TSIG, we have decided to remove the relevant code completely in PowerDNS Authoritative 4.4.0. Users of earlier versions that rely on the feature can keep doing so until they upgrade to 4.4.0, but need to be aware of these issues.

We would like to thank Nathaniel Ferguson for finding and subsequently reporting these issues!

20.22 PowerDNS Security Advisory 2021-01: Specific query crashes Authoritative Server

- CVE: CVE-2021-36754
- Date: July 26th, 2021

- Affects: PowerDNS Authoritative version 4.5.0
- Not affected: 4.4.x and below, 4.5.1
- Severity: High
- Impact: Denial of service
- Exploit: This problem can be triggered via a specific query packet
- Risk of system compromise: None
- Solution: Upgrade to 4.5.1, or filter queries in `dnsdist`

PowerDNS Authoritative Server 4.5.0 (and the alpha/beta/rc1/rc2 prereleases that came before it) will crash with an uncaught out of bounds exception if it receives a query with QTYPE 65535. The offending code was not present in earlier versions, and they are not affected.

Users that cannot upgrade immediately, but do have dnsdist in place, can use dnsdist to filter such queries before they do harm, with something like `addAction(QTypeRule(65535), RCodeAction(DNSRCode.REFUSED))`.

When the PowerDNS Authoritative Server is run inside a supervisor like supervisord or systemd, an uncaught exception crash will lead to an automatic restart, limiting the impact to a somewhat degraded service.

We would like to thank Reinier Schoof and Robin Geuze of TransIP for noticing crashes in production, immediately letting us know, and helping us figure out what was happening.

20.23 PowerDNS Security Advisory 2022-01: incomplete validation of incoming IXFR transfer in Authoritative Server and Recursor

- CVE: CVE-2022-27227
- Date: 25th of March 2022.
- Affects: PowerDNS Authoritative version 4.4.2, 4.5.3, 4.6.0 and PowerDNS Recursor 4.4.7, 4.5.7 and 4.6.0
- Not affected: PowerDNS Authoritative Server 4.4.3, 4.5.4, 4.6.1 and PowerDNS Recursor 4.4.8, 4.5.8 and 4.6.1
- Severity: Low
- Impact: Denial of service
- Exploit: This problem can be triggered by an attacker controlling the network path for IXFR transfers
- Risk of system compromise: None
- Solution: Upgrade to patched version, do not use IXFR in Authoritative Server
- In the Authoritative server this issue only applies to secondary zones for which IXFR transfers have been enabled and the network path to the primary server is not trusted. Note that IXFR transfers are not enabled by default.
- In the Recursor it applies to setups retrieving one or more RPZ zones from a remote server if the network path to the server is not trusted.

IXFR usually exchanges only the modifications between two versions of a zone, but sometimes needs to fall back to a full transfer of the current version. When IXFR falls back to a full zone transfer, an attacker in position of man-in-the-middle can cause the transfer to be prematurely interrupted. This interrupted transfer is mistakenly interpreted as a complete transfer, causing an incomplete zone to be processed. For the Authoritative Server, IXFR transfers are not enabled by default. The Recursor only uses IXFR for retrieving RPZ zones. An incomplete RPZ transfer results in missing policy entries, potentially causing some DNS names and IP addresses to not be properly intercepted.

We would like to thank Nicolas Dehaine and Dmitry Shabanov from ThreatSTOP for reporting and initial analysis of this issue.

20.24 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.

CHAPTER

TWENTYONE

CHANGELOGS

The changelogs for the PowerDNS Authoritative Server are split between release trains.

21.1 Changelogs for 4.9.x

21.1.1 4.9.3

Released: 17th of December 2024

This is release 4.9.3 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

Bug Fixes

- SVCB parser: allow quotes around port= ¶ References: pull request 14968

21.1.2 4.9.2

Released: 1st of October 2024

This is release 4.9.2 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

New Features

- provide dns packet when doing lookup for presigned signatures ¶ References: pull request 14712
- auth web: make request/response timeout configurable (Samir Aguiar) ¶ References: pull request 14245

Improvements

- Yahttp router: appease coverity regarding unsigned underflow in match() ¶ References: pull request 14710
- Fix build with boost 1.86.0 ¶ References: pull request 14709
- more and better thread names ¶ References: pull request 14707
- webserver.cc: try to pass exception text with 500 errors ¶ References: pull request 14706
- ssqlite3: use "begin immediate" to avoid later fights over upgrading to write ¶ References: pull request 14603
- switch el7 builds to Oracle Linux 7 ¶ References: pull request 14403
- dns.cc: use `pdns::views::UnsignedCharView` ¶ References: pull request 14405

- geoipbackend: link C++ filesystem lib if necessary ¶ References: pull request 14303

Bug Fixes

- pdnsutil check-zone: make LUA records optional ¶ References: pull request 14711
- json11: add include for cstdint ¶ References: pull request 14708
- API: when querying with rrset_name, respect it for comments too ¶ References: pull request 14675
- fix domain ordering in catalog SQL query ¶ References: pull request 14526

21.1.3 4.9.1

Released: 28th of May 2024

This is release 4.9.1 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

Improvements

- rpm: Change home directory to /var/lib/pdns ¶ References: pull request 14188
- m4: Add option for 64-bit time_t on 32-bit systems with glibc-2.34 (Sven Wegener) ¶ References: pull request 14191
- Wrap backend factories in smart pointers ¶ References: pull request 14187
- (optionally) drop whitespace on join ¶ References: pull request 14126

Bug Fixes

- autoconf: allow prerelease systemd versions (Chris Hofstaedtler) ¶ References: pull request 14253
- ixfrdist: Fix broken 'uid' and 'gid' parsing for non-numerical values ¶ References: pull request 14251
- YaHTTP: Enforce max # of request fields and max request line size ¶ References: pull request 14229
- Fix memory leaks in the bind file format parser ¶ References: pull request 14199
- dnsproxy: fix build on s390x (Chris Hofstaedtler) ¶ References: pull request 14186
- pdnsutil check-zone: accept LUA A/AAAA as SVCB address targets ¶ References: pull request 14189
- Properly finalize PKCS11 modules before releasing them (Aki Tuomi) ¶ References: pull request 14190
- don't crash when a catalog SOA is invalid ¶ References: pull request 14058
- debian: adjust option name in shipped postinst ¶ References: pull request 13929

21.1.4 4.9.0

Released: 15th of March 2024

This is release 4.9.0 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

4.9 contains improvements to the API, ALIAS handling, catalog zones, and some tool improvements. It also contains various bug fixes and smaller improvements, please see the list below.

New Features

- LUA dblookup: switch qtype argument to int ¶ References: pull request 13900
- LUA: support returning empty set in filterForward #13879 ¶ References: pull request 13895

Improvements

- on OpenBSD, try harder to send on a non-blocking socket ¶ References: pull request 13901
- Docker: Only print config if debug flag is set ¶ References: pull request 13898

Bug Fixes

- revive remotebackend tests and fix failures ¶ References: pull request 13899
- do not disable ns records at apex in consumer zones ¶ References: pull request 13897
- catalog: include groups in hash calculation ¶ References: pull request 13896

21.1.5 4.9.0-beta2

Released: 16th of February 2024

This is release 4.9.0-beta2 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

4.9 contains improvements to the API, ALIAS handling, catalog zones, and some tool improvements. It also contains various bug fixes and smaller improvements, please see the list below.

New Features

- ixfrdist: add support for outgoing notify ¶ References: pull request 13753
- LUA records, pickchashed function ¶ References: pull request 13752
- Add Lua function to pick records via name hash (Brian Rak) ¶ References: pull request 13391
- LUA records: add dblookup function ¶ References: pull request 12359
- add a configurable delay for notifications ¶ References: pull request 13649
- Add and document a *localwho()* function for LUA records (Bert Hubert) ¶ References: pull request 13481

Improvements

- API: reject priority element in record ¶ References: pull request 13743
- dnsname: Optimize parsing of uncompressed labels ¶ References: pull request 13724
- Log port with all freshness check failure scenarios. (Sander Smeenk) ¶ References: pull request 13110
- DNSName: correct len and offset types ¶ References: pull request 13723
- getAllDomains catalog: avoid useless copy ¶ References: pull request 13729
- LUA createForward: allow non-hex word prefix ¶ References: pull request 13722

Bug Fixes

- lmdb: remove mapasync mode, it was always a lie ¶ References: pull request 13803
- debian: adjust option names in shipped configs ¶ References: pull request 13738
- fix tinydnsbackend compilation issue ¶ References: pull request 13725
- set catalog in gsql getAllDomains ¶ References: pull request 13633

21.1.6 4.9.0-beta1

Released: not released

21.1.7 4.9.0-alpha1

Released: 12th of January 2024

This is release 4.9.0-alpha1 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.9.x.

This version contains improvements to the API, ALIAS handling, catalog zones, and some tool improvements. It also contains various bug fixes and smaller improvements, please see the list below.

New Features

- forward EDNS Client Subnet option during ALIAS processing ¶ References: pull request 13441
- add loglevel-show setting to get logs formatted like structured logs ¶ References: pull request 13062
- ixfrdist: add NOTIFY receive support ¶ References: pull request 13322
- dnsscope: Add a *–port* option to select a custom port ¶ References: pull request 13287
- sdig: add rudimentary EDE output ¶ References: pull request 13293
- add default-catalog-zone setting ¶ References: pull request 13238
- API: replace zone contents et al (Chris Hofstaedtler) ¶ References: pull request 12086
- geoipbackend: Support reading zones from directory (Aki Tuomi) ¶ References: pull request 11597

Improvements

- iputils: avoid unused warnings on !linux ¶ References: pull request 13693
- Remove the *extern'ed 'StatBag* from *ws-auth* ¶ References: pull request 13613
- allow building in separate build directory (Chris Hofstaedtler) ¶ References: pull request 13642
- Move method checking to Router (Aki Tuomi) ¶ References: pull request 13153, pull request 13641
- Add supervisor to Auth container image ¶ References: pull request 13619
- Remove legacy terms from the codebase (Kees Monshouwer) ¶ References: pull request 13023
- Wrap `DIR*` objects in unique pointers to prevent memory leaks ¶ References: pull request 13191
- bindparser add primary/secondary/etc. keywords (Kees Monshouwer) ¶ References: pull request 13028
- Netmask: Normalize subnet masks coming from a string ¶ References: pull request 13340
- Report auth settings deprecated in 4.5 (Josh Soref) ¶ References: pull request 13014
- Improve error message for missing GSS-TSIG feature (Andreas Jakum) ¶ References: pull request 13192

- Print the list of loaded modules next to the config.h preset ¶ References: pull request 13162
- Change the default for building with net-snmp from *auto* to *no* ¶ References: pull request 13168
- harmonize *xfr log messages (Josh Soref) ¶ References: pull request 12565
- Refactor the MultiThreadDistributor using *pdns::channel* ¶ References: pull request 12949
- report which backend failed to instantiate ¶ References: pull request 13065
- add remote to logs when tcp thread dies (Chris Hofstaedtler) ¶ References: pull request 13063
- Add missing tools to pdns-tools package description (control) (Andreas Jakum) ¶ References: pull request 13049
- pkcs11signers: If private key object has *CKA_ALWAYS_AUTHENTICATE* attribute, perform *CKU_CONTEXT_SPECIFIC* login after *OperationInit* to make it actually work. (Aki Tuomi) ¶ References: pull request 12753
- wait for *mysql.service* (Andras Kovacs) ¶ References: pull request 13029
- bump sdist builders to alpine 3.18 ¶ References: pull request 12877
- new option 'ignore-errors' for setting 'outgoing-axfr-expand-alias' (Klaus Darilion) ¶ References: pull request 11510

Bug Fixes

- improve wildcard CNAME handling (Kees Monshouwer) ¶ References: pull request 13635
- auth api: flush all caches when flushing (Chris Hofstaedtler) ¶ References: pull request 13514
- CAA records: handle empty value more gracefully, fixes #13070 ¶ References: pull request 13072
- calidns: Fix setting an ECS source of 0 ¶ References: pull request 13018
- calidns: Prevent a crash on an empty domains file ¶ References: pull request 13019

21.2 Changelogs for 4.8.x

21.2.1 4.8.4

Released: 21st of December 2023

This is release 4.8.4 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

New Features

- Add supervisor to Auth container image ¶ References: pull request 13625

Bug Fixes

- extend the systemd startup timeout during lmdb schema migrations ¶ References: pull request 13626
- ixfrdist: Fix the validation of 'max-soa-refresh' ¶ References: pull request 13624

21.2.2 4.8.3

Released: 5th of October 2023

This is release 4.8.3 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

This release contains one new feature (`default-catalog-zone`), one bugfix (in ixfrdist), and a workaround for a bug in the MySQL client libraries.

New Features

- add default-catalog-zone setting ¶ References: pull request 13240

Improvements

- smysql: stop explicitly setting MYSQL_OPT_RECONNECT to 0 ¶ References: pull request 13271

Bug Fixes

- ixfrdist: set AA=1 on SOA responses ¶ References: pull request 13316

21.2.3 4.8.2

Released: 7th of September 2023

This is release 4.8.2 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

This release contains a small collection of fixes:

Bug Fixes

- (I)XFR: handle partial read of len prefix ¶ References: pull request 13186
- fix code producing json ¶ References: pull request 13187
- calidns: fix setting an ECS source of 0 ¶ References: pull request 13188
- Fix incorrect optsize ¶ References: pull request 13189
- lmdb: when broadcasting indexes, -do- rewrite them even if they are unchanged ¶ References: pull request 13099

21.2.4 4.8.1

Released: 7th of July 2023

This is release 4.8.1 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

This release contains a small collection of fixes:

New Features

- lmdb: add backend commands for checking & refreshing indexes ¶ References: pull request 12997

Improvements

- lmdb: in Lightning Stream mode, during deleteDomain, use RW transaction to get ID list ¶ References: pull request 12996
- Stop using the now deprecated ERR_load_CRYPTO_strings() to detect OpenSSL ¶ References: pull request 12993
- Work around Red Hat 8 pooping the bed in OpenSSL's headers ¶ References: pull request 12991

Bug Fixes

- YaHTTP: Prevent integer overflow on very 3large chunks ¶ References: pull request 12992

21.2.5 4.8.0

Released: 1st of June 2023

This is release 4.8.0 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

In 4.8, the LMDB backend gains a new Lightning Stream-compatible schema, which requires a data migration (this is automatic, and there is no migration back to the old schema). LMDB backend users should pay extra attention to the *Upgrade Notes*.

Lightning Stream is an open source data syncer that allows multiple nodes to sync LMDB (Lightning Memory-Mapped Database) data to and from an S3 (compatible) bucket. This has particular advantages in distributed and/or large-scale applications (i.e. ~1 million records), making DNS replication much, much easier to manage.

We are excited about how Lightning Stream simplifies running multiple distributed PowerDNS Authoritative servers, with full support for keeping record data and DNSSEC keys in sync, from multiple writers.

4.8.0 improves the handling of accidental duplicate domains – deleting a zone now deletes all versions of it. This release also contains a few other fixes, please see the list below.

Bug Fixes

- do not answer with broken TYPE0 data when expanding an ENT wildcard ¶ References: pull request 12869
- lmdb: delete duplicate domain entries in deleteDomain ¶ References: pull request 12872
- pdnsutil: if user pushes unknown key in response to "problem with zone" prompt, do not throw away their changes ¶ References: pull request 12868
- add setting workaround-11804-max-chunk-records ¶ References: pull request 12828

21.2.6 4.8.0-beta1

Released: 4th of May 2023

This is release 4.8.0-beta1 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

In 4.8, the LMDB backend gains a new Lightning Stream-compatible schema, which requires a data migration (this is automatic, and there is no migration back to the old schema). LMDB backend users should pay extra attention to the *Upgrade Notes*.

Lightning Stream is an open source data syncer that allows multiple nodes to sync LMDB (Lightning Memory-Mapped Database) data to and from an S3 (compatible) bucket. This has particular advantages in distributed and/or large-scale applications (i.e. ~1 million records), making DNS replication much, much easier to manage.

We are excited about how Lightning Stream simplifies running multiple distributed PowerDNS Authoritative servers, with full support for keeping record data and DNSSEC keys in sync, from multiple writers.

4.8.0-beta1 adds logic to deal with domains existing twice in the database when two Lightning Stream nodes manage to add it at the same time. It also contains a few other fixes, please see the list below.

New Features

- ixfrdist: add a per domain max-soa-refresh option ¶ References: pull request 12768
- LUA records: enhance ifportup() with lists of sets of addresses like ifurlup() ¶ References: pull request 12669

Improvements

- lmdb: handle lack of support for RRset comments better ¶ References: pull request 12636
- calm down the communicator loop (Kees Monshouwer) ¶ References: pull request 12721
- DNSRecord: Ensure that the content can be read or replaced, not edited ¶ References: pull request 12664

Bug Fixes

- LMDB: handle duplicate domain existence consistently ¶ References: pull request 12729
- Pick the right signer name when a NSEC name is also a delegation point (Kees Monshouwer) ¶ References: pull request 12740
- Fixes a typo in pdnsutil clear-zone help output (san983) ¶ References: pull request 12706

21.2.7 4.8.0-alpha1

Released: 21st of March 2023

This is release 4.8.0-alpha1 of the Authoritative Server.

Please review the *Upgrade Notes* before upgrading from versions < 4.8.x.

In this release, the LMDB backend gains a new Lightning Stream-compatible schema, which requires a data migration (this is automatic, and there is no migration back to the old schema). LMDB backend users should pay extra attention to the *Upgrade Notes*.

Lightning Stream is an open source data syncer that allows multiple nodes to sync LMDB (Lightning Memory-Mapped Database) data to and from an S3 (compatible) bucket. This has particular advantages in distributed and/or large-scale applications (i.e. ~1 million records), making DNS replication much, much easier to manage.

We are excited about how Lightning Stream simplifies running multiple distributed PowerDNS Authoritative servers, with full support for keeping record data and DNSSEC keys in sync, from multiple writers.

This version also contains various other small fixes and improvements. Except for very minor ones, they are listed below.

New Features

- new lmdbbackend storage schema that is compatible with Lightning Stream ¶ References: pull request 12546
- add byteslimit support to lua ifurlup() function ¶ References: pull request 12277
- pdnsutil, implement list-member-zones (Kees Monshouwer) ¶ References: pull request 12406
- LUA: accept more hex formats in createForward[6] ¶ References: pull request 12349

Improvements

- cleanup for OpenSSL 3.0 API ¶ References: pull request 11600, pull request 12401, pull request 12414, pull request 12423, pull request 12462, pull request 12501, pull request 12502, pull request 12513, pull request 12515, pull request 12516, pull request 12524, pull request 12527, pull request 12540, pull request 12550
- service files: Add more sandboxing options ¶ References: pull request 11065
- move alsoNotifies up into DNSBackend ¶ References: pull request 12265
- Basic abstraction for handling colored terminal output, respecting isatty(), –no-colors and NO_COLOR ¶ References: pull request 11346
- change sdig output order (Kees Monshouwer) ¶ References: pull request 11858

Bug Fixes

- save errno value as close(2) might clobber it ¶ References: pull request 11101
- Fix multiple-version IXFR request handling in ixfrdist (Håkan Lindqvist) ¶ References: pull request 12127
- Properly encode json strings containing binary data ¶ References: pull request 12260
- lmdb, fix TSIG key removal (Kees Monshouwer) ¶ References: pull request 12322
- Update supported record types: remove A6 MAILA MAILB SIG and WKS, add ZONEMD ¶ References: pull request 12252
- Detect invalid bytes in *makeBytesFromHex()* ¶ References: pull request 12066
- initialize zone cache after dropping privileges (Kees Monshouwer) ¶ References: pull request 11908
- Prevent a race during the processing of SVC auf-hints ¶ References: pull request 12659
- fix ColumnSize argument in SQLBindParameter #12324 (v1shnya) ¶ References: pull request 12331

21.3 Changelogs for 4.7.x

21.3.1 4.7.4

Released: 17th of April 2023

This is release 4.7.4 of the Authoritative Server. It contains various bug fixes, some performance improvements, and one new feature (`pdnsutil list-member-zones`).

New Features

- pdnsutil, implement list-member-zones (Kees Monshouwer) ¶ References: pull request 12676

Improvements

- lock.hh: include <stdexcept> ¶ References: pull request 12458
- calm down the communicator loop ¶ References: pull request 12745
- timeout handling for IXFRs as a client ¶ References: pull request 12743

Bug Fixes

- Properly encode json string containing binary data ¶ References: pull request 12742
- Prevent a race during the processing of SVC auto-hints ¶ References: pull request 12741
- lmdb delete fixes and tests (Kees Monshouwer) ¶ References: pull request 12675
- minicurl: stop leaking hostlist memory ¶ References: pull request 12429
- ixfrdist fixes and improvements ¶ References: pull request 12521
- Pick the right signer name when a NSEC name is also a delegation point ¶ References: pull request 12746
- Fix multiple-version IXFR request handling in ixfrdist ¶ References: pull request 12744

21.3.2 4.7.3

Released: 9th of December 2022

This is release 4.7.3 of the Authoritative Server. It contains various fixes.

Improvements

- API: slightly clearer message when a backend cannot create domains ¶ References: pull request 12296

Bug Fixes

- lmdb: make outgoing notifications work ¶ References: pull request 12299
- lmdb: implement alsoNotifies ¶ References: pull request 12266
- API: do not create SOA and NS records for consumer zones ¶ References: pull request 12291
- API: fix newly created zone not rectified ¶ References: pull request 12273
- fix invalid catalog zone sql query for gpgsqlbackend ¶ References: pull request 12272
- fix pdns_control list-zones ¶ References: pull request 12181

21.3.3 4.7.2

Released: 1st of November 2022

This is version 4.7.2 of the Authoritative Server. It fixes one bug spotted after the release of 4.7.0, for which we forgot to include the fix in 4.7.1.

Bug Fixes

- Un-reverse xfr freshness check ¶ References: pull request 12130

21.3.4 4.7.1

Released: 31st of October 2022

This is version 4.7.1 of the Authoritative Server. It fixes a few bugs spotted after the release of 4.7.0.

Bug Fixes

- include auth 4.7 schema upgrade files in tarballs and packages ¶ References: pull request 12110
- catalog zones: avoid bulk zone reset while migrating to a catalog (Kees Monshouwer) ¶ References: pull request 12124
- catalog zones: stop wasting options update queries (Kees Monshouwer) ¶ References: pull request 12124

21.3.5 4.7.0

Released: 20th of October 2022

This is version 4.7.0 of the Authoritative Server.

4.7.0 brings support for *Catalog Zones*, developed by Kees Monshouwer. As part of that development, the freshness checks in the Primary code were reworked, reducing them from doing potentially thousands of SQL queries (if you have thousands of domains) to only a few. Installations with lots of domains will benefit greatly from this, even without using catalog zones.

4.7.0 also brings back GSS-TSIG support, previously removed for quality reasons, now reworked with many stability improvements.

Other things of note:

- LUA records, when queried over TCP, can now re-use a Lua state, giving a serious performance boost.
- lmdbbackend databases now get a UUID assigned, making it easy for external software to spot if a database was completely replaced
- lmdbbackend databases now optionally use random IDs for objects
- a new LUA function called `ifurlextup`, and improvements in other LUA record functions
- autoprimary management in `pdnsutil` and the HTTP API
- in beta, a key roller daemon, currently not packaged

Please make sure to read the *upgrade notes* before upgrading.

Besides that, various other smaller features and improvements have landed - please browse the list below.

Improvements

- pdnsutil check-zone, skip metadata check for backends without getAllDomainMetadata() (Kees Monshouwer) ¶ References: pull request 12085

Bug Fixes

- Fix compilation of the event ports multiplexer (Jonathan Perkin) ¶ References: pull request 12069
- fix axfr for tinydns and pipe backend (Kees Monshouwer). Note that this was only broken since 4.7.0-beta2. ¶ References: pull request 12098

21.3.6 4.7.0-rc1

Released: 3rd of October 2022

This is the first release candidate for Authoritative Server 4.7.0.

4.7.0 brings support for *Catalog Zones*, developed by Kees Monshouwer. As part of that development, the freshness checks in the Primary code were reworked, reducing them from doing potentially thousands of SQL queries (if you have thousands of domains) to only a few. Installations with lots of domains will benefit greatly from this, even without using catalog zones.

4.7.0 also brings back GSS-TSIG support, previously removed for quality reasons, now reworked with many stability improvements.

Other things of note:

- LUA records, when queried over TCP, can now re-use a Lua state, giving a serious performance boost.
- lmdbbackend databases now get a UUID assigned, making it easy for external software to spot if a database was completely replaced
- lmdbbackend databases now optionally use random IDs for objects
- a new LUA function called `ifurlextup`, and improvements in other LUA record functions
- autoprimary management in `pdnsutil` and the HTTP API
- in beta, a key roller daemon, currently not packaged

Please make sure to read the *upgrade notes* before upgrading.

Besides that, various other smaller features and improvements have landed - please browse the list below.

New Features

- add keyroller ¶ References: pull request 12042

Improvements

- pdnsutil edit-zone, detect capitalization changes in LUA, TXT and SPF records (Kees Monshouwer) ¶ References: pull request 12040
- clang14 has reached macOS ¶ References: pull request 12029
- docker: upgrade to bullseye ¶ References: pull request 11972

Bug Fixes

- AXFR server: abort on chunk with TC set ¶ References: pull request 12043
- axfr-retriever: abort on chunk with TC set ¶ References: pull request 12030

21.3.7 4.7.0-beta2

Released: 13th of September 2022

This is the first published beta for Authoritative Server 4.7.0. (beta1 was never released because of bugs found during the release process).

4.7.0 brings support for *Catalog Zones*, developed by Kees Monshouwer. As part of that development, the freshness checks in the Primary code were reworked, reducing them from doing potentially thousands of SQL queries (if you have thousands of domains) to only a few. Installations with lots of domains will benefit greatly from this, even without using catalog zones.

4.7.0 also brings back GSS-TSIG support, previously removed for quality reasons, now reworked with many stability improvements.

Other things of note:

- LUA records, when queried over TCP, can now re-use a Lua state, giving a serious performance boost.
- lmdbbackend databases now get a UUID assigned, making it easy for external software to spot if a database was completely replaced

- lmdbbackend databases now optionally use random IDs for objects
- a new LUA function called `ifurlextup`, and improvements in other LUA record functions
- autoprimary management in `pdnsutil` and the HTTP API

Please make sure to read the *upgrade notes* before upgrading.

Besides that, various other smaller features and improvements have landed - please browse the list below.

New Features

- Implement catalog zones in the authoritative server (Kees Monshouwer) ¶ References: pull request 11772, pull request 11822, pull request 11825, pull request 11836
- Extend LUA records (rage4) ¶ References: pull request 11588
- RSA, ECDSA PEM import/export ¶ References: pull request 11414, pull request 11426, pull request 11432, pull request 11493, pull request 11590
- auth API: fetch individual rrsets ¶ References: pull request 11389
- reintroduce GSS-TSIG support ¶ References: pull request 11143
- New setting compare-signatures-on-zone-freshness-check to disable DO flag for SOA checks ¶ References: pull request 11508

Improvements

- some small NSEC3PARAM-related fixes to the REST API (Kees Monshouwer) ¶ References: pull request 11918
- LUA records: make shared mode work for TCP queries ¶ References: pull request 11760, pull request 11929, pull request 11933
- Also allow generic record format in zone parsing for pdnsutil zonemd-verify-file ¶ References: pull request 11727
- pdnsutil flush prompt (norve) ¶ References: pull request 11340
- Change dns_tolower() and dns_toupper() to use a table ¶ References: pull request 11655
- auth packaging: add DoT support to sdig ¶ References: pull request 11639
- Tweak for Coverity 1488422 ¶ References: pull request 11599
- Try harder to find libdecaf headers ¶ References: pull request 11562
- lmdb: make map size configurable ¶ References: pull request 11328
- web: stop sending Server: header ¶ References: pull request 11813
- libssl: Properly load ciphers and digests with OpenSSL 3.0 ¶ References: pull request 11862

Bug Fixes

- use getInnerRemote() for the remotes ring (Kees Monshouwer) ¶ References: pull request 11842
- make sure a notified zone is in the zone cache (Kees Monshouwer) ¶ References: pull request 11815
- getTSIGKey(s) cleanup (Kees Monshouwer) ¶ References: pull request 11755, pull request 11759
- fix deleteDomain() in lmdb backend (Kees Monshouwer) ¶ References: pull request 11764
- 2136: match autosplit TXT more usefully ¶ References: pull request 11738
- no ALIAS and LUA record expansion in presigned zones (Kees Monshouwer) ¶ References: pull request 11350

- ixfr: Fix a case where an incomplete read caused by network error might result in a truncated zone ¶ References: pull request 11466
- fix proxy protocol query statistics (Kees Monshouwer) ¶ References: pull request 11314
- lmdb random-ids: stop generating negative numbers ¶ References: pull request 11354
- Log "NULL" for nullptr-bound properties instead of dereferencing ¶ References: pull request 11882
- initialize zone cache after dropping privileges ¶ References: pull request 11908
- Fix libcrypto handling in automake files ¶ References: pull request 11860

21.3.8 4.7.0-beta1

Released: never

Bugs were found after beta1 was tagged.

21.3.9 4.7.0-alpha1

Released: 17th of February 2022

This is version 4.7.0-alpha1 of the Authoritative Server. This release contains a few new features compared to 4.6, and a couple of other fixes/changes.

New features:

- lmdbbackend databases now get a UUID assigned, making it easy for external software to spot if a database was completely replaced
- lmdbbackend databases now optionally use random IDs for objects
- a new LUA function called `ifurlextup`
- autoprimary management in `pdnsutil` and the HTTP API

Please make sure to read the *upgrade notes* before upgrading.

New Features

- lmdb: add random ID generation feature ¶ References: pull request 11309
- auth LUA: add ifurlextup function ¶ References: pull request 11227
- auth lmdb: add a UUID to newly created databases ¶ References: pull request 11241
- new `pdnsutil zonemd-verify-file` command ¶ References: pull request 11100
- auth: add autoprimary management in API & pdnsutil ¶ References: pull request 11102

Improvements

- el7 builds: switch to boost 1.69 ¶ References: pull request 11315
- builder: migrate EL8 builds to oraclelinux, rename centos8 to el8 where possible ¶ References: pull request 11262
- libssl: fix compilation issues on older openssl versions ¶ References: pull request 11108

Bug Fixes

- lmdb: default values for KeyDataDB members, thanks ubsan ¶ References: pull request 11306
- save errno value as close(2) might clobber it ¶ References: pull request 11101

21.4 Changelogs for 4.6.x

21.4.1 4.6.4

Released: 9th of December 2022

This is release 4.6.4 of the Authoritative Server. It contains various small fixes.

Improvements

- docker: upgrade to bullseye ¶ References: pull request 11978

Bug Fixes

- pdnsutil edit-zone, detect capitalization changes in LUA, TXT and SPF records ¶ References: pull request 12303
- initialize zone cache after dropping privileges; Detect invalid bytes in makeBytesFromHex() (Kees Monshouwer) ¶ References: pull request 11945
- Log "NULL" for nullptr-bound properties instead of dereferencing ¶ References: pull request 12035
- LUA records: we only need one IsUpOracle checker thread ¶ References: pull request 12033
- axfr-retriever: abort on chunk with TC set ¶ References: pull request 12031
- use getInnerRemote() for the remotes ring ¶ References: pull request 11871

21.4.2 4.6.3

Released: 13th of July 2022

This is version 4.6.3 of the Authoritative Server. It contains a few bug fixes, and marks the appearance of Ubuntu Jammy packages for the 4.6 branch.

Bug Fixes

- fix deleteDomain() in lmdb backend (Kees Monshouwer) ¶ References: pull request 11765
- RFC2136: match autosplit TXT correctly ¶ References: pull request 11746

21.4.3 4.6.2

Released: 12th of April 2022

This is version 4.6.2 of the Authoritative Server. It contains a carefully selected set of new features, plus a few bug fixes.

New Features

- API: fetch individual rrsets ¶ References: pull request 11409
- LUA: add ifurlextup function ¶ References: pull request 11408

Improvements

- LMDB backports:
  - each LMDB database now gets a UUID
  - lmdbbackend can now (optionally: *lmdb-random-ids*) use random IDs instead of incremental IDs for objects
  - LMDB map size is now configurable (*lmdb-map-size*)
  - one uninitialised memory issue that was fixed

  ¶ References: pull request 11406

Bug Fixes

- fix proxy protocol query statistics and add more detailed latency metrics ¶ References: pull request 11407

21.4.4 4.6.1

Released: 25th of March 2022

This is a security fix release for *PowerDNS Security Advisory 2022-01*. Additionally, because CentOS 8 is End Of Life now, we have switched those builds to Oracle Linux 8. The resulting packages are compatible with RHEL and all derivatives.

Bug Fixes

- Fix validation of incremental zone transfers (IXFRs). ¶ References: pull request 11455

21.4.5 4.6.0

Released: 25th of January 2022

This is version 4.6.0 of the Authoritative Server.

This release contains a ton of improvements and bug fixes compared to 4.5, and a couple of user visible changes.

New features:

- support for incoming PROXY headers
- support for EDNS Cookies
- autoprimary management via pdnsutil and the API

Please make sure to read the *upgrade notes* before upgrading.

Compared to 4.6.0-rc1, this release contains a couple bug fixes and a small improvement.

Improvements

- show QType by name in log message (Chris Hofstaedtler) ¶ References: pull request 11206

Bug Fixes

- Fix swagger/openapi spec for new autoprimary management interface ¶ References: pull request 11205
- fix multiple CDS records in an AXFR (Kees Monshouwer) ¶ References: pull request 11220

21.4.6 4.6.0-rc1

Released: 14th of January 2022

This is version 4.6.0-rc1 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.5, and a couple of user visible changes.

New features:

- support for incoming PROXY headers
- support for EDNS Cookies
- autoprimary management via pdnsutil and the API

Please make sure to read the *upgrade notes* before upgrading.

Compared to 4.6.0-beta1, this release contains one new feature (autoprimary management via API and pdnsutil) and a bunch of bug fixes.

New Features

- add autoprimary management in API & pdnsutil ¶ References: pull request 11161

Improvements

- add zone removal to the zone cache (Kees Monshouwer) ¶ References: pull request 11160
- docker images: Remove capability requirements ¶ References: pull request 11093

Bug Fixes

- pdnsutil edit-zone: fix n and e behaviour on increase-serial prompt ¶ References: pull request 11145
- lmdb: check if the lookup name is part of the zone (Kees Monshouwer) ¶ References: pull request 11159
- lmdb: fix records removal in deleteDomain(); improve tcp exception handling ¶ References: pull request 11139

21.4.7 4.6.0-beta1

Released: 9th of December 2021

This is version 4.6.0-beta1 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.5, and a couple of user visible changes.

New features:

- support for incoming PROXY headers
- support for EDNS Cookies

Please make sure to read the *upgrade notes* before upgrading.

Compared to 4.6.0-alpha1, the major user visible change is the new NSEC3PARAM settings (again, see the upgrade docs). Besides that, see below for various fixes.

Improvements

- pdnsutil: skip SHA1 DSes except in verbose mode ¶ References: pull request 11067
- add zone to zone cache on flush API endpoint ¶ References: pull request 11066
- new default nsec3param settings ¶ References: pull request 11012
- api, check qtype location (Kees Monshouwer) ¶ References: pull request 10959
- put some json on /api/v1 ¶ References: pull request 10865
- Add more UDP error metrics (checksum, IPv6) ¶ References: pull request 10919
- ZoneParserTNG: Stricter checks when loading a zone file ¶ References: pull request 10901
- Implement fd-usage metric for OpenBSD ¶ References: pull request 10891

Bug Fixes

- prevent bind-hybrid+zone-cache ¶ References: pull request 11068
- Condition to HAVE_SYSTEMD_WITH_RUNTIME_DIR_ENV is reversed. During build, the runtime directory in the service files for virtual-hosting are now correctly generated. ¶ References: pull request 10982
- dnsupdate/2136: apply new TTL to whole RRset, not only to the added record ¶ References: pull request 10981
- bindbackend: skip rejected zones during list and search ¶ References: pull request 10941
- make the zone cache more robust for bad data and save some SOA queries for DNSSEC zones (Kees Monshouwer) ¶ References: pull request 10963
- improve dnsname exception handling for SOA records (Kees Monshouwer) ¶ References: pull request 10951

21.4.8 4.6.0-alpha1

Released: 7th of October 2021

This is version 4.6.0-alpha1 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.5, and a couple of user visible changes.

New features:

- support for incoming PROXY headers
- support for EDNS Cookies

Please make sure to read the *upgrade notes* before upgrading.

New Features

- Implement EDNS cookies ¶ References: pull request 10289
- incoming PROXY support ¶ References: pull request 10660

Improvements

- SVCB: Accept known SVCParams in generic format ¶ References: pull request 10727
- Convert locks to LockGuarded ¶ References: pull request 10653
- Move to hashed passwords for the web interface ¶ References: pull request 10157

- remove randombackend (Kees Monshouwer) ¶ References: pull request 10484
- remove attodot feature; fixes #10254 ¶ References: pull request 10780
- auth NOTIFY: log names of NSes we could not resolve ¶ References: pull request 10562
- pdns_control man page: fix copy+paste for decrypt (Josh Soref) ¶ References: pull request 10779
- Gracefully handle uncaught exceptions in the UDP path ¶ References: pull request 10770
- add cookie counters (Kees Monshouwer) ¶ References: pull request 10749

Bug Fixes

- lmdb-safe: resizing while there might be open transactions is unsafe ¶ References: pull request 10672
- checkKey: handle NULL error string from OpenSSL more gracefully ¶ References: pull request 10642
- improve SOA parse exception handling (Kees Monshouwer) ¶ References: pull request 10788
- try to reload rejected zones in bind-backend (Kees Monshouwer) ¶ References: pull request 10748
- send YXDOMAIN for too long DNAME synth ¶ References: pull request 10231

21.5 Changelogs for 4.5.x

21.5.1 4.5.5

Released: 9th of December 2022

This is release 4.5.5 of the Authoritative Server. It contains various small fixes.

Improvements

- docker: upgrade to bullseye ¶ References: pull request 11979

Bug Fixes

- axfr-retriever: abort on chunk with TC set ¶ References: pull request 12032
- LUA records: we only need one IsUpOracle checker thread ¶ References: pull request 12034
- IXFR-in: Fix a case where an incomplete read caused by network error might result in a truncated zone ¶ References: pull request 11454

21.5.2 4.5.4

Released: 25th of March 2022

This is a security fix release for *PowerDNS Security Advisory 2022-01*. Additionally, because CentOS 8 is End Of Life now, we have switched those builds to Oracle Linux 8. The resulting packages are compatible with RHEL and all derivatives.

Bug Fixes

- Fix validation of incremental zone transfers (IXFRs). ¶ References: pull request 11454

21.5.3 4.5.3

Released: 21th of January 2022

This is release 4.5.3 of the Authoritative Server. It mostly contains several robustness fixes for the LMDB backend, and for the zone cache. Please see the full list of fixes here:

Improvements

- 2136: improve some log messages ¶ References: pull request 11002

Bug Fixes

- lmdb, check if the lookup name is part of the zone (Kees Monshouwer) ¶ References: pull request 11158
- pdnsutil edit-zone: fix n and e behaviour on increase-serial prompt ¶ References: pull request 11146
- improve tcp exception handling (Kees Monshouwer) ¶ References: pull request 11123
- lmdb: fix records removal in deleteDomain() (Kees Monshouwer) ¶ References: pull request 11123
- 2136: apply new TTL to whole RRset, not only to the added record ¶ References: pull request 11002

21.5.4 4.5.2

Released: 10th of November 2021

This is release 4.5.2 of the Authoritative Server. It contains several robustness fixes for the bindbackend, and for SOA handling. These fixes are especially important for zone cache users.

Bug Fixes

- bindbackend: skip rejected zones during list and search ¶ References: pull request 10968
- make the zone cache more robust for bad data and save some SOA queries for DNSSEC zones (Kees Monshouwer) ¶ References: pull request 10964
- api, check SOA location (Kees Monshouwer) ¶ References: pull request 10962
- improve dnsname exception handling for SOA records (Kees Monshouwer) ¶ References: pull request 10952
- improve SOA parse exception handling (Kees Monshouwer) ¶ References: pull request 10792
- try to reload rejected zones in bind-backend once every bind-check-interval (Kees Monshouwer) ¶ References: pull request 10778

21.5.5 4.5.1

Released: 26th of July 2021

This is release 4.5.1 of the Authoritative Server. It is strictly a security fix release for *Advisory 2021-01*.

Bug Fixes

- auth: correct upper bounds on d_qtypecounters ¶ References: pull request 10611

21.5.6 4.5.0

Released: 13th of July 2021

This is release 4.5.0 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.4, but very few user visible changes.

There are two notable new features:

- The "zone cache", which allows PowerDNS to keep a list of zones in memory, updated periodically. With this cache, PowerDNS can avoid hitting the database with queries for unknown domains. In some setups, and some attack scenarios, this can make a serious performance difference. Users of backends with dynamically generated zones may want to disable this or at least read the upgrade notes extremely carefully. Many thanks to Chris Hofstaedtler for implementing this. This work by Chris was supported by RcodeZero DNS.
- Priority ordering in the AXFR queue in PowerDNS running as a secondary. Some users with a lot of domains (>100k) sometimes found real changes waiting behind signature refreshes on Thursdays. With the new ordering, those real changes can "skip the line" and get deployed on your secondaries faster. Many thanks to Robin Geuze of TransIP for implementing this.

Since 4.5.0-beta1, the zone cache is enabled by default.

Please make sure to read the *upgrade notes* before upgrading.

Bug Fixes

- fix building without sqlite (this got broken between RC1 and RC2). Thanks to our trusty FreeBSD port maintainer Ralf van der Enden for noticing and reporting this. ¶ References: pull request 10579

21.5.7 4.5.0-rc2

Released: 6th of July 2021

This is the second, and hopefully last, release candidate for version 4.5.0 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.4, but very few user visible changes.

There are two notable new features:

- The "zone cache", which allows PowerDNS to keep a list of zones in memory, updated periodically. With this cache, PowerDNS can avoid hitting the database with queries for unknown domains. In some setups, and some attack scenarios, this can make a serious performance difference. Many thanks to Chris Hofstaedtler for implementing this.
- Priority ordering in the AXFR queue in PowerDNS running as a secondary. Some users with a lot of domains (>100k) sometimes found real changes waiting behind signature refreshes on Thursdays. With the new ordering, those real changes can "skip the line" and get deployed on your secondaries faster. Many thanks to Robin Geuze for implementing this.

Since 4.5.0-beta1, the zone cache is enabled by default.

Please make sure to read the *upgrade notes* before upgrading.

Bug Fixes

- bindbackend: purge caches on zone reload; store nsec3 settings at zone load ¶ References: pull request 10552
- Use correct TTL when caching responses from backends (Robin Geuze) ¶ References: pull request 10551

21.5.8 4.5.0-rc1

Released: 25th of June 2021

This is the first release candidate for version 4.5.0 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.4, but very few user visible changes.

The one notable feature is the "zone cache", which allows PowerDNS to keep a list of zones in memory, updated periodically. With this cache, PowerDNS can avoid hitting the database with queries for unknown domains. In some setups, and some attack scenarios, this can make a serious performance difference.

Since 4.5.0-beta1, the zone cache is enabled by default.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- newCAFromRaw(): create ComboAddress from raw 4/16 byte strings, plus test ¶ References: pull request 9474

Improvements

- 2136: allow placing DNSKEY/CDS/CDNSKEY regardless of direct-dnskey setting ¶ References: pull request 10514
- pdnsutil add-autoprimary: print error when exiting with 1 ¶ References: pull request 10512
- pdnsutil create-zone: better error if default-soa-content is broken ¶ References: pull request 10509
- pdnsutil add-zone-key: clarify ZSK default ¶ References: pull request 10373

Bug Fixes

- SVCB additional processing: delay inserts to avoid invalidating iterator; do not chase chains outside of zone ¶ References: pull request 10522
- pdnsutil edit-zone: correctly reask inc-serial question ¶ References: pull request 10513
- SVCB: on parse error, throw instead of truncate ¶ References: pull request 10511
- SVCB: Fix auto hints removing non-auto hints ¶ References: pull request 10510

21.5.9 4.5.0-beta1

Released: 9th of June 2021

This is version 4.5.0-beta1 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.4, but very few user visible changes.

The one notable feature is the "zone cache", which allows PowerDNS to keep a list of zones in memory, updated periodically. With this cache, PowerDNS can avoid hitting the database with queries for unknown domains. In some setups, and some attack scenarios, this can make a serious performance difference.

In beta1, the zone cache is enabled by default.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- LUA records: add filterForward function, to limit the scope of createForward[6] ¶ References: pull request 10463
- add/fix getAllDomains() and enable the zone cache by default (Kees Monshouwer) ¶ References: pull request 10454

Improvements

- simplify createDomain() (Kees Monshouwer) ¶ References: pull request 10461
- SVCB: rename echconfig to ech and add test vectors from draft ¶ References: pull request 10342

21.5.10 4.5.0-alpha1

Released: 27th of May 2021

This is version 4.5.0-alpha1 of the Authoritative Server. This release contains a ton of improvements and bug fixes compared to 4.4, but very few user visible changes.

The one notable feature is the "zone cache", which allows PowerDNS to keep a list of zones in memory, updated periodically. With this cache, PowerDNS can avoid hitting the database with queries for unknown domains. In some setups, and some attack scenarios, this can make a serious performance difference.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- `SVCB` improvements, including a new `svc-autohints` setting ¶ References: pull request 10074
- New RRtypes supported: `CSYNC`, `NID`, `L32`, `L64`, and `LP` ¶ References: pull request 10078, pull request 10121, pull request 10172, pull request 10234, pull request 10256
- API HTTP cryptokeys: add cds array when configured to do so ¶ References: pull request 10217
- sdig: DoT support; TCP Fast Opens support for TCP/DoT/DoH ¶ References: pull request 10060, pull request 10149, pull request 9995
- Serve NSEC3PARAM when asked without DO ¶ References: pull request 9958
- Add a cache of all zones, avoiding backend lookups for zones that do not exist, and for non-existing subzones. (Chris Hofstaedtler) ¶ References: pull request 10432, pull request 9464

Removed Features

- gpgsql backend: drop refcursor support (it never worked anyway) ¶ References: pull request 10259
- remove local-ipv6, query-local-address6, after their deprecation in 4.4 ¶ References: pull request 10251
- Check `sizeof(time_t)` to be at least 8. This makes it easier for us to handle times beyond the years 2038 and 2106 safely. This removes support for platforms where `time_t` is still only 32 bits wide. ¶ References: pull request 10010

Improvements

- Lower max-nsec3-iterations to 100 (Kees Monshouwer) ¶ References: pull request 10260
- add an option to in/exclude disabled zones in the pdnsutil list-all-zone and list-keys output (Kees Monshouwer) ¶ References: pull request 10421

- Swagger/OpenAPI improvements (Kevin Fleming) ¶ References: pull request 8999, pull request 9788
- Ensure socket-dir matches runtimedir on old systemd ¶ References: pull request 9574
- Various logging improvements (Kees Monshouwer, nzlosh) ¶ References: pull request 9764, pull request 9847, pull request 9848, pull request 9910
- Various improvements to the Docker image (rytis, james-crowley) ¶ References: pull request 10028, pull request 10067, pull request 10068, pull request 10165, pull request 9752, pull request 9803
- Build improvements (support for new compilers and boost versions, etc.), improved usage of some library constructs, and architecture specific fixes ¶ References: pull request 10016, pull request 10137, pull request 10141, pull request 10216, pull request 10245, pull request 10269, pull request 10271, pull request 10310, pull request 10329, pull request 10336, pull request 10344, pull request 9749, pull request 9819, pull request 9831, pull request 9832, pull request 9857, pull request 9876, pull request 9895, pull request 9911, pull request 9914, pull request 9920, pull request 9930, pull request 9932, pull request 9937, pull request 9955, pull request 9979
- Switch to C++17 ¶ References: pull request 9913
- LMDB improvements (better transaction safety; support for the `disabled` field; better upgrade handling; stale reader cleanup; other bug fixes) (Robin Geuze, Kees Monshouwer) ¶ References: pull request 10013, pull request 10099, pull request 10107, pull request 10186, pull request 9885, pull request 9888, pull request 9933
- Implement priority levels in the AXFR queue (Robin Geuze) ¶ References: pull request 10196
- pdns.conf, pdnsutil, pdns_control: add modern aliases for words like master and slave. Add a setting to ignore unknown settings, to make mixed-version testing easier. (Chris Hofstaedtler, Kees Monshouwer)

  While changing names, Kees Monshouwer also renamed 'domain' to 'zone' in a ton of places. ¶ References: pull request 10430, pull request 9658, pull request 9669
- When rectifying, do not update ordernames/auth when there is no need (Kees Monshouwer) ¶ References: pull request 10236
- memory usage reporting: use RES instead of "data" size ¶ References: pull request 10161
- pdnsutil: Warn on CNAME targets for NS, MX and SRV ¶ References: pull request 10087
- Also disable PMTU for IPv6 (it was disabled for IPv4 already) ¶ References: pull request 10264
- Make check-zone also check whether there are duplicate key value pair metadatas for the zone (RobinGeuze) ¶ References: pull request 8813
- detect possible metadata cache pollution (Kees Monshouwer) ¶ References: pull request 9520
- change the consistent-backends default to 'yes' ¶ References: pull request 10401

Bug Fixes

- Make sure we recheck failed SOA lookups for notifies (Kees Monshouwer) ¶ References: pull request 10399
- geoip: set netmask on all string formatting types ¶ References: pull request 9813
- fix rounding inaccuracy in latency statistics (Kees Monshouwer) ¶ References: pull request 9768
- pdnsutil add-record: notice when backend does not support replaceRRSet ¶ References: pull request 9775
- Fixed bugs in the implementations of the `SVCB`, `HTTPS`, `IPSECKEY` and `APL` types. ¶ References: pull request 9766, pull request 9844, pull request 9919
- ALIAS: Ensure A and AAAA are in the NSEC bitmap ¶ References: pull request 10155
- pdnsutil load-zone: reject zones with broken rrs ¶ References: pull request 10081
- pdnsutil edit-zone: do not exit on ZoneParser exception ¶ References: pull request 9826

- fix tcp answer counters (Kees Monshouwer) ¶ References: pull request 10007
- run deleteDomain() inside a transaction (Kees Monshouwer) ¶ References: pull request 10037
- gsqlite3: handle escaping correctly for API search ¶ References: pull request 8829
- fix direct-dnskey in AXFR-out (Kees Monshouwer) ¶ References: pull request 9872
- auth: Don't choke on non-base64 values when importing zone keys ¶ References: pull request 10364
- gpgsql: use SELECT .. RETURNING to get inserted row ID ¶ References: pull request 10392

21.6 Changelogs for 4.4.x

21.6.1 4.4.3

Released: 25th of March 2022

This is a security fix release for *PowerDNS Security Advisory 2022-01*. Additionally, because CentOS 8 is End Of Life now, we have switched those builds to Oracle Linux 8. The resulting packages are compatible with RHEL and all derivatives.

Bug Fixes

- Fix validation of incremental zone transfers (IXFRs). ¶ References: pull request 11453

21.6.2 4.4.2

Released: 25th of November 2021

This is version 4.4.2 of the Authoritative Server. It fixes one issue.

Bug Fixes

- RFC2136/nsupdate: apply new TTL to whole RRset, not only to the added record ¶ References: pull request 11004

21.6.3 4.4.1

Released: 8th of February 2021

This is version 4.4.1 of the Authoritative Server. This releases fixes several small issues discovered since the release of 4.4.0.

Improvements

- debian packaging update ¶ References: pull request 9965
- dockerfiles: do not claim equivs-dummy is built from the pdns source package ¶ References: pull request 9953
- Fix missing #include for gcc-11 ¶ References: pull request 9952
- lmdb: Do a mdb_readers_check to clean up stale readers on database load ¶ References: pull request 9946

Bug Fixes

- fix TCP answer counters ¶ References: pull request 10008
- run deleteDomain() inside a transaction ¶ References: pull request 10039
- lmdb: do not reuse backend that has seen corrupted data ¶ References: pull request 9985
- lmdb: serialise LMDBBackend construction to ensure only a single schema upgrade is attempted ¶ References: pull request 9949
- backport some asan/ubsan fixes ¶ References: pull request 9923
- pdnsutil edit-zone: do not exit on ZoneParser exception ¶ References: pull request 9912

21.6.4 4.4.0

Released: 18th of December 2020

This is version 4.4.0 of the Authoritative Server.

This release drops GSS/TSIG support, please see *PowerDNS Security Advisory 2020-06*.

Version 4.4.0 brings a bunch of exciting changes:

- the LMDB backend now supports long record content, making it production ready for everybody
- the SVCB and HTTPS record types are supported, with limited additional processing
- transaction handling in the 2136 handler and the HTTP API was again improved a lot, avoiding various spurious issues users may have noticed if they do a lot of changes
- a new setting (*consistent-backends*) offers a roughly 30% speedup, subject to conditions
- we finally emit Prometheus metrics!

We want to specifically thank Robin Geuze, Kees Monshouwer, Mischan Toosarani-Hausberger, Chris Hofstaedtler, and Kevin Fleming for their contributions to this release. We are also grateful to all other reporters of bugs, issues, feature requests, and submitters of smaller fixes and features.

Please make sure to read the *upgrade notes* before upgrading.

Improvements

- don’t log trusted-notification-proxy notify at error level (Kees Monshouwer) ¶ References: pull request 9866
- Stop using incbin and use od & sed to generate constant string data. ¶ References: pull request 9853

Bug Fixes

- clear the LMDB set state when performing a new lookup or list to prevent corruption cases (Robin Geuze) ¶ References: pull request 9884
- SVCB: Correctly parse and print unknown params ¶ References: pull request 9878
- fix direct-dnskey in AXFR-out (Kees Monshouwer) ¶ References: pull request 9873

21.6.5 4.4.0-rc1

Released: 7th of December 2020

This is the first Release Candidate for version 4.4.0 of the Authoritative Server. If no trouble surfaces, we will release the actual 4.4.0 within a few weeks.

This release drops GSS/TSIG support, please see *PowerDNS Security Advisory 2020-06*.

Version 4.4.0 brings a bunch of exciting changes:

- the LMDB backend now supports long record content, making it production ready for everybody
- the SVCB and HTTPS record types are supported, with limited additional processing
- transaction handling in the 2136 handler and the HTTP API was again improved a lot, avoiding various spurious issues users may have noticed if they do a lot of changes
- a new setting (*consistent-backends*) offers a roughly 30% speedup, subject to conditions
- we finally emit Prometheus metrics!

We want to specifically thank Robin Geuze, Kees Monshouwer, Mischan Toosarani-Hausberger, Chris Hofstaedtler, and Kevin Fleming for their contributions to this release. We are also grateful to all other reporters of bugs, issues, feature requests, and submitters of smaller fixes and features.

Please make sure to read the *upgrade notes* before upgrading.

Improvements

- pdnsutil add-record: notice when backend does not support replaceRRSet ¶
- add remote to default axfr logging (Kees Monshouwer) ¶ References: pull request 9765
- Fix the DNSName move assignment operator ¶ References: pull request 9761

Bug Fixes

- geoip: set netmask on all string formatting types (Kees Monshouwer) ¶ References: pull request 9816
- api-swagger.json: do not depend on .venv target ¶ References: pull request 9800
- Ensure socket-dir matches runtime dir on old systemd ¶ References: pull request 9798
- fix rounding inaccuracy in latency statistics (Kees Monshouwer) ¶ References: pull request 9785
- APL records: fix endianness problem ¶ References: pull request 9773

21.6.6 4.4.0-beta1

Released: 23rd of November 2020

This is version 4.4.0-beta1 of the Authoritative Server.

This release drops GSS/TSIG support, please see *PowerDNS Security Advisory 2020-06*.

Version 4.4.0 brings a bunch of exciting changes:

- the LMDB backend now supports long record content, making it production ready for everybody
- the SVCB and HTTPS record types are supported, with limited additional processing
- transaction handling in the 2136 handler and the HTTP API was again improved a lot, avoiding various spurious issues users may have noticed if they do a lot of changes
- a new setting (*consistent-backends*) offers a roughly 30% speedup, subject to conditions
- we finally emit Prometheus metrics!

We want to specifically thank Robin Geuze, Kees Monshouwer, Mischan Toosarani-Hausberger, Chris Hofstaedtler, and Kevin Fleming for their contributions to this release. We are also grateful to all other reporters of bugs, issues, feature requests, and submitters of smaller fixes and features.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- allow ip ranges as trusted-notification-proxy ¶ References: pull request 9713
- Add '/api/docs' endpoint to Auth server (Kevin Fleming) ¶ References: pull request 8911
- geoipbackend: accept custom lookup mapping (criteo-forks) ¶ References: pull request 8608
- make it possible to replace now supported TYPExx records (Kees Monshouwer) ¶ References: pull request 9693
- Add pdns_control command to the list of XFR domains in queue ¶ References: pull request 9239

Improvements

- nxdomain performance (Kees Monshouwer) ¶ References: pull request 9704
- logging: put quotes around some IPs to make messages easier to read ¶ References: pull request 9731
- Use Python 3 in build system (Kevin Fleming) ¶ References: pull request 9712
- adjust AXFR, IXFR events loglevels (Kees Monshouwer) ¶ References: pull request 9688
- pdnsutil zone key improvements (Chris Hofstaedtler) ¶ References: pull request 9692
- add some missing counters. Fixes #9675 ¶ References: pull request 9685

Bug Fixes

- pdnsutil check-zone: DNAME fixes ¶ References: pull request 9735
- lmdb: fill di.serial (this fixes the 'serial=0' API bug) ¶ References: pull request 8707
- disable mysql automatic charset detection (see *upgrade notes*) ¶ References: pull request 9714
- tinydnsbackend: Ignore duplicate SOA in getAllDomains() ¶ References: pull request 9174

21.6.7 4.4.0-alpha3

Released: 5th of November 2020

This is version 4.4.0-alpha3 of the Authoritative Server.

Alpha 2 was not released due to the LMDB encoding bug mentioned below, found shortly after starting the Alpha 2 release process.

This release drops GSS/TSIG support, please see *PowerDNS Security Advisory 2020-06*.

Version 4.4.0 brings a bunch of exciting changes:

- the LMDB backend now supports long record content, making it production ready for everybody
- the SVCB and HTTPS record types are supported, with limited additional processing
- transaction handling in the 2136 handler and the HTTP API was again improved a lot, avoiding various spurious issues users may have noticed if they do a lot of changes
- a new setting (*consistent-backends*) offers a roughly 30% speedup, subject to conditions
- we finally emit Prometheus metrics!

We want to specifically thank Robin Geuze, Kees Monshouwer, Mischan Toosarani-Hausberger, and Chris Hofstaedtler for their contributions to this release. We are also grateful to all other reporters of bugs, issues, feature requests, and submitters of smaller fixes and features.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- auth: add support for dnssec removal to CDS/CDNSKEY (Kees Monshouwer) ¶ References: pull request 9631
- pdns_control: add show <prefix> (Chris Hofstaedtler) ¶ References: pull request 9549

Removed Features

- Auth: remove SOA autofilling, remove set-ptr feature from API (Kees Monshouwer) ¶ References: pull request 9593

Improvements

- When *consistent-backends* is enabled, use ANY queries toward backends whenever possible. (Kees Monshouwer) ¶ References: pull request 9483, pull request 9656
- Deprecate *local-ipv6* and *query-local-address6*, to prepare for removal in 4.5.0 (Chris Hofstaedtler, Kees Monshouwer) ¶ References: pull request 9552, pull request 9625
- pdns: bind-backend speedup feedRecord() (Kees Monshouwer) ¶ References: pull request 9611
- auth: Speedup presigned signature lookups. (Kees Monshouwer) ¶ References: pull request 9568
- auth: bindbackend: 'rediscover' changes to 'type' (Roald Stolte) ¶ References: pull request 9645
- gsql,bind: allow seamless serving of newly-supported TYPExx records (Chris Hofstaedtler) ¶ References: pull request 9623

Bug Fixes

- LMDB: Fix encoding of deleted RRsets (Kees Monshouwer) ¶ References: pull request 9664, pull request 9665
- auth lmdb: fill di.backend in getUnfreshSlaveInfos and getAllDomains (this makes the right serial appear in API calls) ¶ References: pull request 9647
- auth: change "misconfigured" SOA MNAME to not mention powerdns and be RFC6761 compliant ¶ References: pull request 9627
- LUA records: handle a potentially uncaught exception ¶ References: pull request 9613
- Fixes for APL records pointing to fe80 (Chris Hofstaedtler) ¶ References: pull request 9550, pull request 9580
- svc-records: Initialize d_port ¶ References: pull request 9544

21.6.8 4.4.0-alpha1

Released: 30th of September 2020

This is version 4.4.0-alpha1 of the Authoritative Server. This release drops GSS/TSIG support, please see *PowerDNS Security Advisory 2020-06*.

Version 4.4.0 brings a bunch of exciting changes:

- the LMDB backend now supports long record content, making it production ready for everybody
- the SVCB and HTTPS record types are supported, with limited additional processing
- transaction handling in the 2136 handler and the HTTP API was again improved a lot, avoiding various spurious issues users may have noticed if they do a lot of changes
- we finally emit Prometheus metrics!

We want to specifically thank Robin Geuze, Kees Monshouwer, Mischan Toosarani-Hausberger, and Chris Hofstaedtler for their contributions to this release. We are also grateful to all other reporters of bugs, issues, feature requests, and submitters of smaller fixes and features.

Please make sure to read the *upgrade notes* before upgrading.

New Features

- Allow forced secondary zone retrieval ¶ References: pull request 8824
- Add a new command to add a super-master to SQL backends (Godwottery) ¶ References: pull request 7963
- API: Allow rectifying Slave zones (Chris Hofstaedtler) ¶ References: pull request 9183
- Implemented prometheus metrics-endpoint for auth (supervacuus) ¶ References: pull request 9182

Removed Features

- Remove GSS/TSIG support ¶ References: pull request 9385

Improvements

- New RRtypes: SVCB, HTTPS, APL. Fixed RRtypes: IPSECKEY. ¶ References: pull request 8638, pull request 9337, pull request 9369
- LMDB: new schema that supports long records (Robin Geuze) ¶ References: pull request 9389
- bindbackend: 'rediscover' changes to master and also-notifies (Matti Hiljanen) ¶ References: pull request 9524
- quote/escape PG connection parameters ¶ References: pull request 9039
- lua: add backtraces to errors ¶ References: pull request 8942
- Skip EDNS Cookies in the packet cache ¶ References: pull request 8993
- Use more of systemd's sandboxing options when available ¶ References: pull request 8969
- auth slave: log successful NOTIFY (Chris Hofstaedtler) ¶ References: pull request 9387
- sdig: Increment the DNS message IDs when pipelining, report ID mismatches ¶ References: pull request 9419, pull request 9430
- Add support for FreeBSD's SO_REUSEPORT_LB ¶ References: pull request 9157
- LUA records: two improvements to createForward ¶ References: pull request 9101
- geoipbackend: top looking after first weighted match, propagate weighted rounding gap fix (criteo-forks) ¶ References: pull request 8564, pull request 8565
- Make a combination of delete and replace for rrset possible (jonathaneen) ¶ References: pull request 8623
- log more pdns_control actions (Chris Hofstaedtler) ¶ References: pull request 9218
- gsqlbackend: allow backend-specific queries (Chris Hofstaedtler) ¶ References: pull request 9318
- add used master address to slave check logs (Chris Hofstaedtler) ¶ References: pull request 9265
- immediately fill account, kind, masters on zone create (Chris Hofstaedtler) ¶ References: pull request 9280
- fetch all metadata at once (Kees Monshouwer) ¶ References: pull request 9169
- Add version 'statistic' to prometheus ¶ References: pull request 9252
- PKCS11 improvements ¶ References: pull request 9215
- gpgsqlbackend: add parameters to query logging (Chris Hofstaedtler) ¶ References: pull request 9189

- Set SyslogIdentifier for multiple instances (Chris Hofstaedtler) ¶ References: pull request 9187
- Optimize IXFR-to-AXFR fallback path (Chris Hofstaedtler) ¶ References: pull request 9163
- auth smysql: mimic error message format from mysql tooling ¶ References: pull request 9024
- improve sql schema updates (Kees Monshouwer) ¶ References: pull request 8975
- NSEC fixes for unpublished DNSKEY (RobinGeuze) ¶ References: pull request 8925, pull request 8939
- make sure we look at 10% of all cached items during cleanup (Kees Monshouwer) ¶ References: pull request 8929
- Reduce the number of temporary memory allocations ¶ References: pull request 8714

Bug Fixes

- Improved transaction handling, especially around the metadata cache (Kees Monshouwer) ¶ References: pull request 9407, pull request 9409, pull request 9427, pull request 9518
- ignore cryptokeys in presigned zones (Kees Monshouwer) ¶ References: pull request 9496
- remove a '// HACK FIXME400' and fix the bugs it was hiding (Kees Monshouwer) ¶ References: pull request 9478
- Fix the sample 'geoip.conf' for Debian-based packages ¶ References: pull request 9439
- Fix building with LLVM11 (RvdE) ¶ References: pull request 9408
- Ensure qtype is set before calling setContent() in axfrfilter() ¶ References: pull request 8928
- Auth API: Allow removal of NSEC3PARAM metadata ¶ References: pull request 9340
- pdnsutil: make sure we let all destructors run. ¶ References: pull request 9253
- Remote Backend: Throw DBException in functions that allow it ¶ References: pull request 8943
- Ensure runtime dirs for virtual services differ ¶ References: pull request 9073
- better (actual) fix for mem leak in SSQLite3::execute() ¶ References: pull request 9080
- Avoid "pthread_rwlock_destroy on rwlock with waiters!" on OpenBSD ¶ References: pull request 9069
- BIND-DOMAIN-EXTENDED-STATUS: don't look for a domain called BIND-DOMAIN-EXTENDED-STATUS ¶ References: pull request 9060

21.7 Changelogs for 4.3.x

21.7.1 4.3.2

Released: 12th of March 2021

This is version 4.3.2 of the Authoritative Server. This release fixes latency calculations to match the approach used in 4.4.0, to make comparisons between 4.3 and 4.4 more useful. It also contains a few build-related improvements.

Bug Fixes

- fix rounding inaccuracy in latency statistics (Kees Monshouwer) ¶ References: pull request 9786

21.7.2 4.3.1

Released: 22nd of September 2020

This is version 4.3.1 of the Authoritative Server. This release contains the fix for *PowerDNS Security Advisory 2020-05* (CVE-2020-17482). It also contains several other fixes and improvements:

New Features

- add ubuntu focal target ¶ References: pull request 9083

Improvements

- EL8 pkgs: Build mysql backend against mariadb-connector-c-devel ¶ References: pull request 9036
- gpgsql: Reintroduce prepared statements (Chris Hofstaedtler) ¶ References: pull request 9219
- gsqlite3backend: add missing indexes (Kees Monshouwer) ¶ References: pull request 9233
- use real remote for supermaster createSlaveDomain() (Kees Monshouwer) ¶ References: pull request 9224
- Optimize IXFR-to-AXFR fallback path (Chris Hofstaedtler) ¶ References: pull request 9176
- Install bind SQL schema files as part of bindbackend (Chris Hofstaedtler) ¶ References: pull request 9013
- Do not send out of zone lookups to the backends (Kees Monshouwer) ¶ References: pull request 9480

Bug Fixes

- Raise an exception on invalid hex content in unknown records. ¶ References: pull request 9498
- Handle the extra single-row result set of MySQL stored procedures (Chris Hofstaedtler) ¶ References: pull request 9444

21.7.3 4.3.0

Released: 7th of April 2020

This is version 4.3.0 of the Authoritative Server. It contains all changes mentioned in the alpha, beta and RC versions below, plus two more bugfixes.

If you are upgrading from beta2 or rc2, AND ONLY THEN, please read pull request #8975 very carefully.

Improvements

- reduce the number of temporary memory allocations ¶ References: pull request 8951

Bug Fixes

- avoid IXFR-in corruption when deltas come in close together (please see the *IXFR-in corruption* upgrade notes) ¶ References: pull request 8977
- improve sql schema updates ¶ References: pull request 8975
- Fix NSECx for unpublished DNSKEYs properly ¶ References: pull request 8973

21.7.4 4.3.0-rc2

Released: 18th of March 2020

This is the first Release Candidate for version 4.3.0 of the Authoritative Server. The version called 4.3.0-rc1 was never released because of the cache cleanup change mentioned below.

Improvements

- Make sure we look at 10% of all cached items during cleanup (Kees Monshouwer) ¶ References: pull request 8924

Bug Fixes

- emit correct NSEC/NSEC3 bitmaps in hidden key situations (Robin Geuze) ¶ References: pull request 8936

21.7.5 4.3.0-beta2

Released: 21st of February 2020

This is the second beta for version 4.3.0 of the Authoritative Server. We expect this to be the final beta. Please see *the upgrade notes* for some breaking changes including a mandatory schema upgrade for database backends.

Some minor fixes have been left out from the list below; some other bugs may have been fixed accidentally as a result of the tremendous amount of work that goes into each of our major releases.

As announced in Backend removals in the upcoming Authoritative Server release, we have removed five backends.

New Features

- add full option to 'pdns_control show-config' (Kees Monshouwer) ¶ References: pull request 8094
- Add 'IO wait' and 'steal' metrics on Linux ¶ References: pull request 8783
- API: add includerings option to statistics endpoint ¶ References: pull request 8784
- Add an extended status report in the bind backend ¶ References: pull request 8682

Improvements

- adjust NSEC TTLs to negative TTL ¶ References: pull request 8811
- Add more SQL schema files to packages and tarballs (Matt Nordhoff) ¶ References: pull request 8776
- only log "No question section in packet" at Debug logging level ¶ References: #7970, pull request 8828
- do not update identical notified serials (Kees Monshouwer) ¶ References: pull request 8096
- IXFR: only sign SOA in empty response for +DO queries ¶ References: pull request 8779
- Prepare the caches' buckets in advance ¶ References: pull request 8793
- Rework NetmaskTree for better CPU and memory efficiency. (Stephan Bosch) ¶ References: pull request 8355

Bug Fixes

- Refuse NSEC records with a bitmap length > 32 ¶ References: pull request 8826
- YaHTTP: Support bracketed IPv6 addresses ¶ References: pull request 8815
- Make sure the default-publish-cds and default-publish-cdnskey options are respected for AXFR (Robin Geuze) ¶ References: pull request 8788

21.7.6 4.3.0-beta1

Released: 31st of January 2020

This is the first beta for version 4.3.0 of the Authoritative Server. Please see *the upgrade notes* for some minor breaking changes.

Some minor fixes have been left out from the list below; some other bugs may have been fixed accidentally as a result of the tremendous amount of work that goes into each of our major releases.

As announced in Backend removals in the upcoming Authoritative Server release, we have removed five backends.

New Features

- add default-publish-{cds|cdnskey} options ¶ References: pull request 8594
- remotebackend: Support alsoNotifies, setFresh, getUnfreshSlaveInfos ¶ References: pull request 8701, pull request 8732
- Add support for managing unpublished DNSSEC keys (Robin Geuze, TransIP) ¶ References: pull request 8177

Improvements

- allow local-ipv6 until 4.4.0 ¶ References: pull request 8749
- Add metrics about the size of our in-memory rings ¶ References: pull request 8681
- gpgsqlbackend: stop using prepared statements (Chris Hofstaedtler) ¶ References: pull request 8627
- Enforce a strict maximum size for the packet and records caches ¶ References: pull request 8713

Bug Fixes

- make sure records from LMDB backend end up in the right packet section (Kees Monshouwer) ¶ References: pull request 8628
- Clear the TSIG algo between iterations in the API ¶ References: pull request 8649
- HTTP API: Allow DNAME in apex with SOA and NS records ¶ References: pull request 8668

misc

- remove the implicit 5->7 algorithm upgrade ¶ References: pull request 8754
- Make Lua mandatory for Auth (Chris Hofstaedtler) ¶ References: pull request 8744

21.7.7 4.3.0-alpha1

Released: 10th of December 2019

> This is the first alpha for version 4.3.0 of the Authoritative Server. Please see *the upgrade notes* for some minor breaking changes.

New Features

- gmysql backend, add an option to send the SSL capability flag (Kees Monshouwer) ¶ References: pull request 8328
- pdnsutil: offer to increase serial after edit-zone ¶ References: pull request 6496

Removed Features

- remove goracle, lua, mydns, opendbx, oracle backends (Kees Monshouwer, Robin Geuze) ¶ References: pull request 7455, pull request 8313, pull request 8432, pull request 8434, pull request 8435, pull request 8436
- deprecate SOA autocomplete in pdnsutil check-zone (Kees Monshouwer) ¶ References: pull request 7918

Improvements

- API: optionally, do not return dnssec info in domain list (zeha) ¶ References: pull request 4628
- zone file parser: Add a parameter to limit the number of '$GENERATE' steps ¶ References: pull request 8451, pull request 8492
- api: avoid a large number of new database connections (Kees Monshouwer) ¶ References: pull request 8457
- Emulate a buffered read in the pipe backend, ~3x faster ¶ References: pull request 8159
- LUA performance: register lua functions only once (chbruyand) ¶ References: pull request 8307
- API: make max request/response body size configurable ¶ References: pull request 7550
- API: add edited_serial to Zone object ¶ References: pull request 7962
- Improve error when notification comes in for non-slave zone ¶ References: pull request 7943
- LUA record: rewrote the health checking system ¶ References: pull request 8249

Bug Fixes

- various memory/thread correctness fixes ¶ References: pull request 8006, pull request 8208, pull request 8350, pull request 8351, pull request 8580, pull request 8601
- LUA view: do not crash on empty IP list ¶ References: #8572, pull request 8575
- REST API: accept headers without spaces ¶ References: pull request 8562
- on luaSynth exception, drain db output ¶ References: #8299, pull request 8301
- tinydnsbackend: limit timestamp-based TTLs (smellyspice) ¶ References: #7439, pull request 8225
- Ensure that pdns can read pdns.conf when upgrading from an older package (mnordhoff) ¶ References: pull request 8352
- Ixfrdist: handle reading of empty files gracefully ¶ References: pull request 8424
- webserver: handle exceptions instead of SIGABRTing the world ¶ References: pull request 8296

21.8 Changelogs for 4.2.x

21.8.1 4.2.3

Released: September 2nd 2020

This release contains the fix for *PowerDNS Security Advisory 2020-05* (CVE-2020-17482)

Bug Fixes

- Raise an exception on invalid hex content in unknown records. ¶ References: pull request 9499
- mydns: add SOA to list() output ¶ References: #9181, pull request 9191

21.8.2 4.2.2

Released: 9th of April 2020

New Features

- api: add includerings option to statistics endpoint ¶ References: pull request 8786

Improvements

- cache: strictly enforce maximum size, and improve cleanup routine ¶ References: pull request 8736, pull request 9003

Bug Fixes

- fix records ending up in wrong packet section (Kees Monshouwer) ¶ References: pull request 9010
- avoid IXFR-in corruption when deltas come in close together (please see the *IXFR-in corruption* upgrade notes) ¶ References: pull request 9001
- fix out-of-bound access for zero length "serialized" string when using lmdbbackend. (Kees Monshouwer) ¶ References: pull request 8612
- bind backend: pthread_mutex_t should be inited and destroyed and not be copied ¶ References: pull request 8602

21.8.3 4.2.1

Released: 2nd of December 2019

This release fixes several bugs and makes a few features more robust or intuitive. It also contains a few performance improvements for API users.

New Features

- Add SLAVE-RENOTIFY zone metadata support (Matti Hiljanen) ¶ References: pull request 8549
- Add configurable timeout for inbound AXFR (Matti Hiljanen) ¶ References: pull request 8547
- Add CentOS 8 as builder target ¶ References: pull request 8428
- gmysql backend, add an option to send the SSL capability flag ¶ References: pull request 8341

Improvements

- API: reduce number of database connections (Kees Monshouwer) ¶ References: pull request 8457
- Register a few known RR types and remove an unknown one ¶ References: pull request 8546
- bindbackend: use metadata for also-notifies as well (Matti Hiljanen) ¶ References: pull request 8548
- pdnsutil increase-serial: under SOA-EDIT=INCEPTION-EPOCH, bump as if it is EPOCH ¶ References: #8218, pull request 8508
- API: optionally do not return dnssec info in domain list (Chris Hofstaedtler) ¶ References: pull request 8541
- Basic validation of $GENERATE parameters ¶ References: pull request 8454

Bug Fixes

- LUA view: do not crash on empty IP list ¶ References: #8572, pull request 8589
- API: Accept headers without spaces ¶ References: pull request 8576
- Avoid database state-related SERVFAILs after a LUA error ¶ References: #8299, pull request 8570
- Just before 4.2.0, some SQL-related fixes broke edit-zone and other features with the LMDB backend. This has been fixed now. (backport by Kees Monshouwer) ¶ References: #8134, pull request 8568
- Clear the caches for the entire zone after a patch operation (was apex only). The default default-api-rectify setting was ignored in patchZone(), rectify only took place when the API-RECTIFY metadata was set to "1". (Kees Monshouwer) ¶ References: pull request 8497
- rfc2136, pdnsutil: somewhat improve duplicate record handling ¶ References: #8217, pull request 8507

21.8.4 4.2.0

Released: 30th of August 2019

Compared to the last release candidate, one more bug has been fixed.

The LMDB backend is incomplete in this version. Slaving zones works, loading zones with pdnsutil works, but more fine-grained edits (using edit-zone, or the REST API) fail. We hope to fix this soon in a 4.2.x release.

For an overview of features new since 4.1.x, please see the 4.2.0 announcement blog post.

Bug Fixes

- bind getAllDomains: ignore per-zone exceptions ¶ References: pull request 8229

21.8.5 4.2.0-rc3

Released: 29th of July 2019

Thanks to an overwhelming amount of testing by our fabulous user community, this release candidate contains a ton of bug fixes (and a few improvements) compared to the previous one. We hope this has shaken out all of the important bugs, so that we can release 4.2.0 soon!

This release, sadly, cripples the LMDB backend somewhat, due to transaction-related fixes for the SQL backends. We hope to fix this issue before 4.2.0, or otherwise, early in 4.2.x.

New Features

- web: make max request/response body size configurable ¶ References: pull request 7550

Improvements

- boost.m4 improvements ¶ References: pull request 8172, pull request 8173
- add metric for open TCP connections ¶ References: pull request 8126
- Various robustness and performance improvements around domain IDs (Kees Monshouwer) ¶ References: pull request 8092
- remove unused import to enable compile on illumos (Thomas Mieslinger) ¶ References: pull request 8064
- ixfrdist: limit XFR chunk size to 16k ¶ References: pull request 8051
- Fix a memory leak when sqlite3_exec() fails ¶ References: pull request 7998
- don't enable the tbhandler when libc only pretends to be glibc (James Taylor) ¶ References: pull request 7980, pull request 8019
- Fix a leak on 'Backend reported permanent error which prevented lookup' error ¶ References: pull request 8006
- Clear CMSG_SPACE(sizeof(data)) in cmsghdr to appease valgrind ¶ References: pull request 7996
- deprecate SOA autocomplete in pdnsutil check-zone (Kees Monshouwer) ¶ References: pull request 7918
- move /var/lib/pdns to pdns-server debian package ¶ References: pull request 7889
- Show newer features in configure output and –version ¶ References: pull request 7890
- completely disable the packet when cache-ttl=0 (Kees Monshouwer) ¶ References: pull request 7910
- Improve error when notification comes in for non-slave zone ¶ References: pull request 7943
- web: add edited_serial to Zone object ¶ References: pull request 7962
- Adapt calidns for openbsd and other systems without rcvmmsg(2) ¶ References: pull request 7871
- DNSName, speed up toString() conversion ¶ References: pull request 7699

Bug Fixes

- packethandler: Compare TSIG key name using DNSName ¶ References: pull request 8168
- Make sure we always compile with BOOST_CB_ENABLE_DEBUG set to 0 ¶ References: pull request 8169
- Fix SERVFAIL when backend returns empty DNSName ¶ References: pull request 8058
- stop using select() in places where FDs can be >1023 ¶ References: pull request 8149
- pdnsutil increase-serial: set right ordername ¶ References: pull request 8166
- use BIGINT for notified_serial in pg schema (Klaus Darilion) ¶ References: pull request 8097
- Fix the accounting of servfail-queries in the distributor ¶ References: pull request 8056
- limit compression pointers to 14 bits ¶ References: pull request 8028
- catch name & IP parse errors during outgoing notify preparations ¶ References: pull request 8037

21.8.6 4.2.0-rc2

Released: 14th of June 2019

Improvements

- Make explicit lmdbbackend synchronous option ¶ References: pull request 7807
- Reduce mmap size for lmdb on 32 bits plus restrict number of shards ¶ References: pull request 7700
- sqlite3: make journal mode configurable; default to WAL ¶ References: pull request 7852
- Portability/building improvements:
  - Update boost.m4 to the latest version
  - Check if `-latomic` is needed instead of hardcoding (neheb)
  - Use `net-snmp-config --netsnmp-agent-libs` instead of `--agent-libs`
  - bump boost requirement to 1.42 unconditionally

  ¶ References: pull request 7668, pull request 7818, pull request 7861, pull request 7862
- Robustness improvements:
  - Fix warnings reported by Coverity
  - Initialize cURL before starting any thread
  - Don't do unaligned memory access

  ¶ References: pull request 7708, pull request 7864, pull request 7865
- Always truncate when the additional records do not fit in a response (Kees Monshouwer) ¶ References: pull request 7873
- Remove `disable-tcp` option ¶ References: pull request 7859
- RKEY is missing algorithm field (DNS-Leo) ¶ References: pull request 7615
- Ignore Path MTU Discovery on UDP server socket ¶ References: pull request 7410
- improve logging in the web server ¶ References: pull request 5932
- GSQL: Log more data in error messages ¶ References: pull request 7584

Bug Fixes

- pdnsutil: show DS for second and further keys too ¶ References: #7667, pull request 7801
- LMDB improvements:
  - lmdbbackend: auth was unset in get() (always true) (Kees Monshouwer)
  - LMDB defaulted to port 0 for master addresses unless explicitly set
  - fix `getAllDomains()` (Kees Monshouwer)

  ¶ References: pull request 7643, pull request 7697, pull request 7784
- auth gsql `getAllDomains`: ignore stou errors ¶ References: pull request 7854
- Rectify/ENT fixes:
  - Allow updates to override existing ENT records
  - Fix ENTs removal when "replacing" new records via the API

  ¶ References: pull request 7787, pull request 7831
- Cleanup SOA editing (Kees Monshouwer) ¶ References: pull request 7363
- DNSSEC fixes:
  - Don't sign insecure records with keys from other zones (Kees Monshouwer)
  - always add DS for secure zones, broken since #7523 (Kees Monshouwer)

¶ References: pull request 7772, pull request 7789

misc

- API: mark `set-ptr` as deprecated (zeha) ¶ References: pull request 7797
- add DoH support to sdig ¶ References: pull request 7832
- dumresp: add TCP support ¶ References: pull request 7655
- pdnsutil, dnswasher: add support for encrypting IP addresses ¶ References: pull request 7481
- auth API, pdnsutil: improve backend transaction correctness ¶ References: pull request 7891
- detect SOA cache pollution caused by broken backends (Kees Monshouwer) ¶ References: pull request 7881
- speedup `getUpdatedMasters()` for the gsql backends (Kees Monshouwer) ¶ References: pull request 7460
- pdns_control reopens geoip databases on reload (jpmens) ¶ References: pull request 7753
- b2b-migrate did not open a transaction, breaking it for lmdb ¶ References: pull request 7696
- No longer filter DNSSEC metadata when DNSSEC is enabled in gsql ¶ References: pull request 7706
- Rectify for ent records in narrow zones was slightly wrong. (Kees Monshouwer) ¶ References: pull request 7580
- Clear caches (meta-data, keys) on domain deletion ¶ References: pull request 7529
- optionally reuse Lua state ¶ References: pull request 7869, pull request 7897
- Various robustness improvements:
  - Do not busy loop if we get lots of notifies.
  - Improve error reporting with garbage in the 'master' field of the database
  - Do not exit on exception resolving addresses to notify
  - Auth ringbuffer summaries were case-sensitive & accounted delegations incorrectly
  - plug mysql_thread_init memory leak
  - Ensure we increase the number of queued queries before decreasing it

  ¶ References: pull request 7503, pull request 7517, pull request 7569, pull request 7587, pull request 7662, pull request 7790
- disable dnssec pre-processing for non dnssec zones and avoid a lot of `isSecuredZone()` calls (Kees Monshouwer) ¶ References: pull request 7523
- rename 'supermaster' option to 'superslave' ¶ References: pull request 7723

21.8.7 4.2.0-rc1

Released: 19th of March 2019

New Features

- Add `dnspcap2calidns` to convert PCAP to the calidns format. ¶ References: pull request 6564
- Add quiet modifier to pdnsutil `rectify-all-zones` command. ¶ References: pull request 6374
- Authoritative LMDB backend. ¶ References: pull request 7453
- Adds the glorious log-log histograms. ¶ References: pull request 6969

- LUA Records (yes we know it is "Lua"). ¶ References: pull request 6171
- LDAP misc updates:
  - ALIAS support,
  - DNAME support.

  ¶ References: #5260, pull request 5821

Removed Features

- Remove `api-logfile` flag and grep API endpoint. ¶ References: pull request 7025
- Drop `api-readonly` configuration setting. ¶ References: pull request 6845

Improvements

- Use a less expensive way to get memory stats for `real-memory-usage`. ¶ References: pull request 7502
- Be smarter about trimming whitespace when creating records from ASCII. ¶ References: pull request 7412
- More sandboxing using SystemD's features. ¶ References: pull request 6634
- Fix attempt to restrict / speed-up additional processing to auth zone. ¶ References: pull request 7353
- Make `pdns_control notify *` also notify slaves zones. ¶ References: pull request 7293
- Zero out QTYPE response numbers in our statistics. Makes Valgrind usable on auth again. ¶ References: pull request 7348
- Improve memory handling for NSEC(3) records with lots of types. ¶ References: pull request 7345
- Use a cache-able soa record for the serial check caused by a notify. ¶ References: pull request 7245
- dns_random: Implement new dns_random. ¶ References: pull request 5274
- Remove `theLog` and `theL` and replace this with a global `g_log`. ¶ References: pull request 6358
- Lower 'packet too short' loglevel. ¶ References: pull request 6312
- Remove all traces of selectmplexer, fix up pollmplexer. ¶ References: pull request 6230
- Change from `time_t` to `uint32_t` for serial in `calculateSOASerial`. ¶ References: #1010, pull request 5068
- Use `toLogString()` for logging and throwing. ¶ References: pull request 5979
- Remove obsolete EDNS PING code. (@zeha) ¶ References: pull request 6156
- Add `type` filter to search-data api. ¶ References: #5430, pull request 7326
- Add `rcode` response statistics on API. ¶ References: #7357, pull request 7359
- Use commas instead of spaces when setting Zone Masters via the REST API. ¶ References: #6451, pull request 7491
- Improve RRset validation. ¶ References: pull request 7463
- Fix a couple of Swagger / OpenAPI issues. ¶ References: pull request 7286
- API: Add TSIG key manipulation endpoints. ¶ References: pull request 5988
- Make API changes do a rectify by default, add an option to disable. ¶ References: pull request 7233
- Add zone lookup by `/zones?zone=example.org`. ¶ References: pull request 6668
- API export function output change to add IN to the output. ¶ References: pull request 6649
- Send correct response codes for the CryptoKey endpoints. ¶ References: #6652, pull request 6662

- Return status 409 if domain already exists. ¶ References: pull request 6325
- Expose ResponseStats via REST API. ¶ References: pull request 2603
- Return 404 for non-existing zones. ¶ References: pull request 6076
- check-all-zones: find duplicate zones and SOAs. ¶ References: pull request 6843
- calidns: Accurate qps targets. ¶ References: pull request 6774
- pdns_control notify: Handle slave w/o renotify properly. ¶ References: pull request 6691
- pdnsutil: Occlusion and auth check improvements. ¶ References: pull request 6653
- pdnsutil: also load modules through the load-modules directive. ¶ References: pull request 6594
- calidns: Add quiet, minimum-success-rate options to use from a script. ¶ References: pull request 6601
- calidns: Add an option to read ECS values from the query file, skip comments. ¶ References: pull request 6526
- calidns: Add a maximum-qps option to stay at a given stable load. ¶ References: pull request 6525
- Add TCP support for ALIAS. ¶ References: pull request 6331
- calidns: Add the --ecs parameter to add random ECS values to queries. ¶ References: pull request 6326
- Add an --initial-port option to dnsreplay. ¶ References: pull request 6166
- Add colour to diff output of pdnsutil. ¶ References: pull request 6063
- remotebackend: Implement getUpdatedMasters. ¶ References: #7444, pull request 7448
- Report checkKey errors upwards. ¶ References: pull request 7516
- ixfrdist: Add option to limit AXFR record count. ¶ References: pull request 6872
- Lua records: Add useragent option to ifurlup and set a default. ¶ References: #7393, pull request 7490
- Lua: Expose dns_random as pdnsrandom. ¶ References: #6853, pull request 7492
- Error on DNSSEC default misconfiguration. ¶ References: pull request 7340
- Fallback to SHA1 for the signatures cache if MD5 is not available. ¶ References: pull request 7284
- Improved Lua records - Added all selector, and backupSelector fallbacks. ¶ References: pull request 6894
- Configure --enable-pdns-option --with-third-party-module. ¶ References: pull request 7026
- Address some known LUA Records issues:
  - Better check input lists,
  - Report lua wildcards errors,
  - Exposes DNSName::getRawLabels in lua env,
  - Better document LUA functions and objects.

  ¶ References: #6693, pull request 6731
- Remove out-of-zone-additional-processing setting. ¶ References: pull request 6869
- Improve RSA key warnings. ¶ References: pull request 6958
- Use unique pointers in the OpenSSL signer. ¶ References: pull request 7069
- Store NetmaskTree nodes in a set for faster removal. ¶ References: pull request 6962
- ALIAS: Respond SERVFAIL on non-NOERRORs from resolver. ¶ References: pull request 6727
- Add support for OpenSSL 1.1.1's ed25519 and ed448 for signing and verifying. ¶ References: pull request 6910

- Add incremental slave-check backoff also for failed AXFR. ¶ References: pull request 6822
- Enhance query-logging with timing for MySQL, PostgreSQL and SQLite. ¶ References: pull request 6975
- Apply ALIAS scopemask after chasing. ¶ References: pull request 6811
- Fix compilation with LibreSSL 2.7.0+. ¶ References: pull request 6948
- Remove GOST and Botan support. ¶ References: pull request 6921
- Add option send-signed-notify to send NOTIFYs without TSIG signature. ¶ References: pull request 6825
- Add the serials when logging the final result of a slave check. ¶ References: pull request 6824
- Implement a smarter dedup for filling packets in auth. ¶ References: pull request 6730
- luawrapper: Report caught std::exception as lua_error. ¶ References: pull request 6658
- Reject duplicate RRsets in patchZone. ¶ References: pull request 6633
- Ensure ALIAS answers over TCP have correct name. ¶ References: pull request 6659
- Fix some minor issues for presigned (large) bind zones. ¶ References: pull request 6561
- bindbackend: Refuse launch suffixes. ¶ References: pull request 6558
- Add missing overrides. ¶ References: pull request 6530
- Avoid an isane amount of new backend connections during an AXFR. ¶ References: pull request 6427
- Add support for MB and MG RR types. ¶ References: pull request 6377
- Add actual EDNS buffer size logging, not just our interpretation. ¶ References: pull request 6102
- Lua2 backend: This is a rewrite of the lua backend. It uses AuthLua4 as basis and more strongly typed access using LuaContext. ¶ References: pull request 6157
- Make requests always return to sender, for usage in multimaster slave zones. Also - made sure that the master that is questioned for updates will be selected randomly, to prevent repeatedly asking a dead master for updates. ¶ References: #3602, pull request 5361
- Reject updates if they would lead to CNAME+Other data. ¶ References: pull request 6276
- Fix rectify (ordername) for non-DNSSEC zones. ¶ References: pull request 6243
- pkcs11signers: Fix yubikey NEO to work. ¶ References: pull request 6278
- Make outgoing-query-address and outgoing-query-address6 behaviours equivalent. ¶ References: pull request 6100
- GeoIPbackend improvements:
  - Adds MMDB support. Now geoip backend can be compiled without geoip support,
  - Adds location support,
  - Fixes SERVFAIL if expansion is empty.

  ¶ References: #5268, pull request 6128
- Fix syntax error for replace-rrset. (@lordievader) ¶ References: pull request 6295
- Add XPF support to sdig, PowerDNS Recursor and dnsdist. ¶ References: #5079, #5594, #5654, pull request 6220
- Check more thoroughly the source of UDP answers. ¶ References: pull request 5960
- Slave cleanups. (@zeha) ¶ References: pull request 6162
- pdns: Improve record parsing ¶ References: #2611, pull request 6158
- Changes to compile and run on NetBSD. ¶ References: pull request 6146
- Forbid creating algo 5/8/10 keys with out-of-spec sizes. ¶ References: pull request 6139

- Add Draft of Swagger spec for Authoritative Server HTTP API. ¶ References: #5854, pull request 5862
- API: Add response-by-qtype and response-by-rcode on /statistics endpoint ¶ References: #7357, pull request 7359
- Several improvements to processing of notifies.
  - Turn off supermaster support by default (adds new setting).
  - PowerDNS was wasting a lot of queries while processing notifies.
  - Use comboaddress for IPs (was strings)

  ¶ References: pull request 6021

Bug Fixes

- Fix `dns_random()` always returning 0 when the minimum acceptable value is 0. ¶ References: pull request 7382
- Lower `udp-truncation-threshold` by default to 1232. ¶ References: pull request 7320
- Prevent leak of file descriptor if running out of ports for incoming AXFR. ¶ References: pull request 7294
- Fix API search failed with "Commands out of sync; you can't run this command now". ¶ References: #7545, pull request 7546
- Improve handling of out of range `modified_at` value. ¶ References: #6114, pull request 7488
- Prevent more than one CNAME / SOA record in the same RRset. ¶ References: #7277, pull request 7278
- Check DNSNames that should be hostnames. ¶ References: pull request 6871
- Make sure that we use strict weak records ordering in the API. (Doing this avoids concurrent records / comments iteration from running out of sync.) ¶ References: pull request 6780, pull request 6816
- Remove ENTs when "replacing" new records. ¶ References: pull request 6647
- Restrict creation of OPT and TSIG rrsets. ¶ References: pull request 6614
- Increase serial after DNSSEC related updates. ¶ References: pull request 6571
- Fix output order of pdnsutil `add-record`. ¶ References: pull request 7482
- sdig: Handle non-IN class records better. ¶ References: pull request 7148
- Dnsscope off-by-one + domain-filter. ¶ References: pull request 7364
- pdns_notify: Support hostname for notification. ¶ References: #7319, pull request 7244
- Make `edit-zone` catch zoneparser exceptions as well. ¶ References: pull request 6859
- calidns: Don't issue socket buffer or SCHED_FIFO warnings in quiet mode. ¶ References: pull request 6617
- dnsreplay: Bail out on a too small outgoing buffer. ¶ References: pull request 6580
- pdnsutil: Use new domain in `b2bmigrate`. ¶ References: pull request 6559
- Link `dnspcap2protobuf` against librt when needed. ¶ References: pull request 6487
- Rather than crash, sheepishly report no file/linenum in pdnsutil. ¶ References: pull request 6354
- IXFR: correct behavior of dealing with DNS Name with multiple records; speed up IXFR transaction. ¶ References: pull request 6172
- Fix invalid SOA record in MySQL which prevented the authoritative server from starting. ¶ References: #7493, pull request 7496
- Avoid infinite loop in mydnsbackend. ¶ References: pull request 7475
- Fix listing zones incl. AXFR. ¶ References: pull request 6122

- Fix static lookup when using weighted records on multiple record types. ¶ References: pull request 7219
- Forbid 0 as weight value. ¶ References: #7219, pull request 7227
- Insufficient validation in the HTTP remote backend (CVE-2019-3871, PowerDNS Security Advisory *2019-03*) ¶ References: #7573, pull request 7576
- Correctly interpret an empty AXFR response to an IXFR query. ¶ References: pull request 7494
- Respect packet size limits, even with ECS and TSIG. ¶ References: pull request 7352
- Fix dot stripping in `setcontent()`. ¶ References: #7429, pull request 7459
- Do not compress the root since LMDB backend cannot set a root zone with a compressible SOA record. ¶ References: #7471, pull request 7472
- Avoid duplicate NSEC3 records in presigned zones in LMDB backend. ¶ References: #7453, pull request 7470
- Fix replying from ANY address for non-standard port. ¶ References: pull request 7341
- Fix a few off-by-one errors. ¶ References: pull request 7288
- On incoming NOTIFY load our serial from backend to have it available during slave-check.

  Also log ourserial to ease debugging. ¶ References: #6821, pull request 6823
- Remove `autoserial` from the Authoritative Server. Serial 0 was a little bit too special in PowerDNS. ¶ References: #1355, #1366, pull request 6838
- Handle ANY queries with Lua records. ¶ References: pull request 7228
- geoip: properly delete libGeoIP return values. ¶ References: pull request 7217
- SOA-check: reject NXDOMAIN response and check label of RR against qname. ¶ References: pull request 7067
- Fix `carbon-instance` / `carbon-namespace` inconsistencies. ¶ References: pull request 7201
- Fix up the BIND config files on upgrade. ¶ References: pull request 7134
- geoipbackend: Handle read error for config file. ¶ References: pull request 7024
- Make sure we escape `127` in TXT records. ¶ References: pull request 7017
- Add support for NONE SOA-EDIT kind. ¶ References: pull request 4598
- Respond correctly to DS query at delegation in unsigned zone. ¶ References: pull request 6923
- Release memory in case of error in the OpenSSL ECDSA constructor. ¶ References: pull request 6917
- Actually truncate truncated responses. ¶ References: #6912, pull request 6913
- RFC2136 fixes. ¶ References: pull request 6858
- Remove SOA-check backoff on incoming NOTIFY and fix `d_lock` handling. ¶ References: pull request 6857
- `check-zone`: allow null MX, SRV. ¶ References: pull request 6834
- Workaround MariaDB pretending to be MySQL. ¶ References: pull request 6844
- Reset the TSIG state between queries. ¶ References: pull request 6738
- Only parse `resolv.conf` once - this avoids race conditions. ¶ References: pull request 6495
- Sign CDS/CDNSKEY RRsets with the KSK. ¶ References: pull request 6655
- Initialize some missed qtypes: WKS, SMIMEA. ¶ References: pull request 6686
- geoipbackend: Check `GeoIP_id_by_addr_gl` and `GeoIP_id_by_addr_v6_gl` return value. ¶ References: pull request 6677
- stubresolver: Improve locking. ¶ References: pull request 6499

- gmysql: Use future-proof statement for transaction isolation. ¶ References: pull request 6648
- Geoip: Fix poisoning of cache when hit service's default network.

  Also includes an optimization to make lookups faster. ¶ References: #6584, pull request 6585
- Avoid interleaved access to B (via `d_dk`). Before this patch, the meta lookup would interfere with the already-started `B.lookup`. This caused failures with odbc/MSSQL. ¶ References: pull request 6531
- Add return 0 for correct exit of `set-kind` and `set-account`. ¶ References: pull request 6481
- Recheck serial when AXFR is done. ¶ References: pull request 6484
- Report unparsable data in stoul `invalid_argument` exception. ¶ References: pull request 6396
- Fix handling of user defined AXFR filters return values. ¶ References: pull request 6370
- Reload `/etc/resolv.conf` when modified. ¶ References: #6263, pull request 6342
- Make `check-zone` error on rows that have content but shouldn't. ¶ References: pull request 6297
- bindbackend: handle `std::exception` during startup zone-parsing. ¶ References: pull request 6152
- gmysql-backend: set unsigned attribute on `notified_serial` column. ¶ References: #5915, pull request 6019
- Escaping unusual DNS label octets in DNSName is off by one. ¶ References: pull request 6018
- Update EDNS Option code list. ¶ References: pull request 6155
- Remove `serializeSOAData`, refactor `calculate`/`edit`/`increaseSOA`. ¶ References: pull request 4547
- Add methods missing from AuthLua4 when Lua support is disabled. ¶ References: pull request 6132
- Init openssl and libsodium before chrooting in pdnsutil. ¶ References: pull request 6129
- Fix uninitialized index in Lua's DNSPacket::getRRS() binding. ¶ References: pull request 6107
- Fix out of bounds exception in CAA processing. ¶ References: #6089, pull request 6103
- Forbid label compression in ALIAS wire format. ¶ References: #6028, pull request 6029

misc

- pdnsutil.1 & settings:
  - Add Ed25519 and Ed448,
  - Document ECC keysizes,
  - Remove old algos.

  ¶ References: pull request 7187
- Name threads in the programs. ¶ References: pull request 6997
- dnsreplay: Add more checks against bogus PCAP. ¶ References: pull request 6582
- geoipbackend: Allow empty content for ENT record. ¶ References: #7195, pull request 7196
- Add namespace and instance variable to carbon key. ¶ References: #2362, #6951, pull request 6959
- LuaWrapper: Disable maybe uninitialized warnings with boost optional. ¶ References: pull request 6769
- Improve tests and two bugfixes:
  - Fix xfrIP to reject invalid ips,
  - Accept seconds since epoch in RRSIG timestamps too.

  (@stbuehler) ¶ References: #6101, #6120, pull request 6124

21.9 Changelogs for 4.1.x

21.9.1 4.1.14

Released: September 2nd 2020

This release contains the fix for *PowerDNS Security Advisory 2020-05* (CVE-2020-17482)

Bug Fixes

- Raise an exception on invalid hex content in unknown records. ¶ References: pull request 9500

21.9.2 4.1.13

Released: August 9th 2019

This is a bugfix release for high traffic setups using the pipebackend or remotebackend.

Bug Fixes

- gpgsqlbackend: add missing schema file to Makefile (tcely) ¶ References: pull request 8157
- stop using select() in places where FDs can be >1023 ¶ References: pull request 8162

21.9.3 4.1.12

Released: 4.1.12 was skipped due to a packaging issue.

21.9.4 4.1.11

Released: August 1st 2019

This release contains the updated PostgreSQL schema for PowerDNS Security Advisory *2019-06* (CVE-2019-10203).

Upgrading is not enough - you need to manually apply the schema change: `ALTER TABLE domains ALTER notified_serial TYPE bigint USING CASE WHEN notified_serial >= 0 THEN notified_serial::bigint END;`

Bug Fixes

- Update PostgreSQL schema for 2019-06. ¶ References: pull request 8144

21.9.5 4.1.10

Released: June 21st 2019

This release and 4.1.9 together fix the following security advisories:

- PowerDNS Security Advisory *2019-04* (CVE-2019-10162)
- PowerDNS Security Advisory *2019-05* (CVE-2019-10163)

Bug Fixes

- Do not exit on exception parsing names of name servers to notify. ¶ References: pull request 7964

21.9.6 4.1.9

Released: June 19th 2019

New Features

- Add an option to disable superslaving. ¶ References: pull request 7922

Bug Fixes

- Do not exit on exception resolving addresses to notify. ¶ References: pull request 7663
- Avoid very busy looping on lots of notifies. ¶ References: pull request 7829
- In gsql `getAllDomainMetadata`, actually get all domain metadata. This makes DNSSEC metadata work with `pdnsutil b2b-migrate`. ¶ References: pull request 7921

21.9.7 4.1.8

Released: March 22nd 2019

Bug Fixes

- Fix rectify for ENT records in narrow zones. ¶ References: #7580, pull request 7609
- Prevent leak of file descriptor if running out of ports for incoming AXFR. ¶ References: #7294, pull request 7603
- EL6: fix `CXXFLAGS` to build with compiler optimizations. ¶ References: pull request 7567
- Fix API search failed with "Commands out of sync; you can't run this command now". ¶ References: #7546, pull request 7602
- Fix invalid SOA record in MySQL which prevented the authoritative server from starting. ¶ References: #7496, pull request 7605
- Plug `mysql_thread_init` memory leak. ¶ References: #7517, pull request 7509
- Correctly interpret an empty AXFR response to an IXFR query. ¶ References: #7494, pull request 7604
- Fix replying from ANY address for non-standard port. ¶ References: #7341, pull request 7610
- Do not compress the root. ¶ References: #7472, pull request 7607
- Fix dot stripping in `setcontent()`. ¶ References: #7459, pull request 7608

21.9.8 4.1.7

Released: March 18th 2019

This release fixes the following security advisory:

- PowerDNS Security Advisory *2019-03* (CVE-2019-3871)

Bug Fixes

- Insufficient validation in the HTTP remote backend (CVE-2019-3871, PowerDNS Security Advisory *2019-03*) ¶ References: pull request 7577

21.9.9 4.1.6

Released: January 31st 2019

Bug Fixes

- Prevent more than one CNAME/SOA record in the same RRset. ¶ References: pull request 7279

21.9.10 4.1.5

Released: November 6th 2018

This release fixes the following security advisories:

- PowerDNS Security Advisory *2018-03* (CVE-2018-10851)
- PowerDNS Security Advisory *2018-05* (CVE-2018-14626)

Improvements

- Apply alias scopemask after chasing ¶ References: pull request 6976
- Release memory in case of error in the openssl ecdsa constructor ¶ References: pull request 6917
- Switch to devtoolset 7 for el6 ¶ References: #7040, pull request 7118

Bug Fixes

- Fix compilation with libressl 2.7.0+ ¶ References: #6943, pull request 6948
- Actually truncate truncated responses ¶ References: pull request 6913
- Crafted zone record can cause a denial of service (CVE-2018-10851, PowerDNS Security Advisory *2018-03*) ¶ References: pull request 7149
- Packet cache pollution via crafted query (CVE-2018-14626, PowerDNS Security Advisory *2018-05*) ¶ References: pull request 7149

21.9.11 4.1.4

Released: August 29th 2018

Improvements

- Fix warnings reported by gcc 8.1.0. ¶ References: pull request 6590
- Make the gmysql backend future-proof. (Chris Hofstaedtler)
  - Use future-proof statement for transaction isolation
  - Allow compiling against MySQL 8 client header files
  - Workaround MariaDB pretending to be MySQL

¶ References: #6632, pull request 6648, pull request 6842, pull request 6844

- Initialize some missed qtypes. (Chris Hofstaedtler) ¶ References: #6685, pull request 6686

Bug Fixes

- Avoid concurrent records/comments iteration from running out of sync. ¶ References: pull request 6780
- Fix a crash in the API when adding records ¶ References: pull request 6816
- `pdns_control notify`: handle slave without renotify properly. (Chris Hofstaedtler) ¶ References: #4457, pull request 6691
- Reset the TSIG state between queries. ¶ References: #6736, pull request 6738
- Remove SOA-check backoff on incoming notify and fix lock handling. (Klaus Darilion) ¶ References: pull request 6857
- Fix an issue where updating a record via DNS-UPDATE in a child zone that also exists in the parent zone, we would incorrectly apply the update to the parent zone. ¶ References: pull request 6858
- Geoipbackend: check `geoip_id_by_addr_gl` and `geoip_id_by_addr_v6_gl` return value. (Aki Tuomi) ¶ References: #6676, pull request 6677

21.9.12 4.1.3

Released: 24th of May 2018

This release contains several small fixes to the GeoIP backend. The most prominent fix being one where the backend would be slow when thousands of network masks were configured for services.

Improvements

- pdnsutil: use new domain in b2bmigrate (Aki Tuomi) ¶ References: #6239, pull request 6559
- Update copyright years to 2018 (Matt Nordhoff) ¶ References: pull request 6130

Bug Fixes

- Restrict creation of OPT and TSIG RRsets ¶ References: #6441, pull request 6614
- Fix handling of user-defined axfr filters return values ¶ References: #6228, pull request 6370
- Prevent the GeoIP backend from copying NetMaskTrees around, fixes slow-downs in certain configurations (Aki Tuomi) ¶ References: #6584, #6585, pull request 6585, pull request 6608
- Ensure alias answers over tcp have correct name ¶ References: #6654, pull request 6659

21.9.13 4.1.2

Released: 8th of May 2018

This is the third release in the 4.1 train. Besides bug fixes, it contains some performance and usability improvements.

Improvements

- API: increase serial after dnssec related updates (Kees Monshouwer) ¶ References: pull request 6571, pull request 6572
- Auth: lower 'packet too short' loglevel ¶ References: #6312, #6545, pull request 6573
- Make check-zone error on rows that have content but shouldn't ¶ References: #6297, pull request 6553
- Auth: avoid an isane amount of new backend connections during an axfr ¶ References: #6427, pull request 6554
- Report unparsable data in stoul invalid_argument exception ¶ References: #6396, pull request 6555
- Backport: recheck serial when axfr is done ¶ References: #6484, pull request 6490
- Backport: add tcp support for alias ¶ References: #6331, pull request 6337

Bug Fixes

- Auth: allocate new statements after reconnecting to postgresql ¶ References: #6464, pull request 6551
- Auth-bindbackend: only compare ips in ismaster() (Kees Monshouwer) ¶ References: #6021, pull request 6570
- Rather than crash, sheepishly report no file/linenum ¶ References: #6354, pull request 6556
- Document undocumented config vars ¶ References: #6539, pull request 6540
- Backport #6276 (auth 4.1.x): prevent cname + other data with dnsupdate ¶ References: #6276, pull request 6314

misc

- Move includes around to avoid boost L conflict ¶ References: #6516, pull request 6542
- Backport: update edns option code list ¶ References: #6155, pull request 6543
- Auth: link dnspcap2protobuf against librt when needed ¶ References: #6487, pull request 6548
- Fix a warning on botan >= 2.5.0 ¶ References: #6478, pull request 6549
- Auth 4.1.x: unbreak build ¶ References: #6304, pull request 6552
- Dnsreplay: bail out on a too small outgoing buffer ¶ References: pull request 6575
- Backport: debian authoritative packaging fixes for 4.1 ¶ References: #6286, pull request 6546
- Fix piddir detection on centos 6 ¶ References: #6051, #6062, pull request 6547

21.9.14 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

21.9.15 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

21.9.16 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: #3059, #5948, #5949, pull request 5879

Bug Fixes

- Use `_exit()` when we *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

21.9.17 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: #3417, #5712, 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: #2250, #5734, #5797, #5889, 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

- 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
- Improve trailing dot handling internally which lead to a segfault in pdnsutil before. ¶ References: #5673, pull request 5684
- 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

21.9.18 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 1701, pull request 4964, pull request 4965
- Add API endpoints for Domain metadata (Christian Kröger). ¶ References: pull request 4093, pull request 5038
- 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 5269, pull request 5270
- 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, #2606, #3337, #4238, #4315, #4616, 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 (Chris 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: #349, #602, pull request 4953
- Remove d_place from DNSResourceRecord (Chris 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' (Chris Hofstaedtler). ¶ References: #4290, pull request 4408
- Prevent sending nameservers list and zone-level NS in rrsets in the API (Chris Hofstaedtler). ¶ References: #4132, pull request 4751
- Forbid mixing CNAMEs and other RRSets in the API (Christan Hofstaedtler). ¶ References: #5305, pull request 5389
- Prevent duplicate records in single RRset (Chris 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: #2358, #5193, 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 (Chris 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, Chris 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 5212, pull request 5249
- 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 5083, pull request 5161
- Zero the port when creating a netmask from a ComboAddress. ¶ References: pull request 5408
- Drop (broken) support for packet-specific SOA replies from backends (Chris 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 (Chris Hofstaedtler). ¶ References: #4524, pull request 4526
- Avoid creating fake DNSPacket objects just for calling getAuth() from API code (Chris 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 (Chris Hofstaedtler). ¶ References: #4328, pull request 4650
- Make statement actually unique (Chris Hofstaedtler). ¶ References: #4928, pull request 4929
- Reconnect to the server if the My/Pg connection has been closed. ¶ References: #3824, #5005, pull request 5245
- 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 ordername 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
- Add missing query for last key insert id in the goracle backend (Aki Tuomi). ¶ References: pull request 5506

21.10 Changelogs for 4.0.x

21.10.1 PowerDNS Authoritative Server 4.0.9

Released 1st of August 2019

This release contains the updated PostgreSQL schema for PowerDNS Security Advisory *2019-06* (CVE-2019-10203).

Upgrading is not enough - you need to manually apply the schema change: `ALTER TABLE domains ALTER notified_serial TYPE bigint USING CASE WHEN notified_serial >= 0 THEN notified_serial::bigint END;`

21.10.2 PowerDNS Authoritative Server 4.0.8

Released 21st of June 2019

This release fixes PowerDNS Security Advisories *2019-04* and *2019-05*.

21.10.3 PowerDNS Authoritative Server 4.0.7

Released 18th of March 2019

This release fixes PowerDNS Security Advisory *2019-03*: Insufficient validation in the HTTP remote backend (CVE-2019-3871)

Bug fixes

- #7582: Insufficient validation in the HTTP remote backend (CVE-2019-3871)

21.10.4 PowerDNS Authoritative Server 4.0.6

Released 6th of November 2018

This release fixes PowerDNS Security Advisory *2018-03*: Crafted zone record can cause a denial of service (CVE-2018-10851)

Bug fixes

- #7150: Crafted zone record can cause a denial of service (CVE-2018-10851)
- #7135: Fix el6 builds

Improvements

- #6315: Prevent cname + other data with dnsupdate
- #7119: Switch to devtoolset 7 for el6

21.10.5 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 (Chris Hofstaedtler)
- #4751: API: prevent sending nameservers list and zone-level NS in rrsets (Chris Hofstaedtler)
- #4929: gpgsql: make statement names actually unique (Chris 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 (Chris 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

21.10.6 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 (Chris 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

21.10.7 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"

21.10.8 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 (Chris 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 3a0bded, 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 (Chris 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 (Chris Hofstaedtler)
- #4754: In `Bind2Backend::lookup()`, use the `zoneId` when we have it

21.10.9 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 > sizeof(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 (Chris Hofstaedtler)
- #4169 Fix typos in a logmessage and exception (Chris 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)

21.10.10 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 ordername (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 and INCEPTION-WEEK 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 gsqlite3: 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

21.10.11 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 (Chris Hofstaedtler)
- #3917 Remotebackend: Add getAllDomains call (Aki Tuomi)

Bug fixes and changes:

- #3998 remove gsql::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` (Chris Hofstaedtler)

21.10.12 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 (Chris 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

21.10.13 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 (Chris 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 (Chris 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 (Chris Hofstaedtler)
- #3632 pdnsutil: Fix exit statuses to constants and return 0 when success (saltsa)
- #3655 API: Fix set-ptr to honor SOA-EDIT-API (Chris 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

21.10.14 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 (Chris 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 (Chris Hofstaedtler)

- #3428 API: Fix zone/records design mistake (Chris 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 (Chris 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 Nelless)
- #3443 Many speedup and correctness fixes

21.10.15 PowerDNS Authoritative Server 4.0.0-alpha1

Released December 24th 2015

21.11 Changelogs for 3.x and older

These changelogs are included for historical purposes. Broken links may exist.

21.11.1 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)

21.11.2 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)

21.11.3 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 (Chris Hofstaedtler)
- commit 8044a5d: Don't reply to truncated queries (Chris 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)

21.11.4 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 dnsdist 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)

21.11.5 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

21.11.6 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)

21.11.7 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)

21.11.8 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 (Chris 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

21.11.9 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 (Chris Hofstaedtler):

- commit ca44706: API: move shared DomainInfo reader into its 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

21.11.10 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.

21.11.11 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.

21.11.12 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 synthesized 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 nameservers share an 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 Chris 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: gpsqlbackend: 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.

21.11.13 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: gpsqlbackend: 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

21.11.14 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 Chris 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

21.11.15 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, 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 ksk|zsk 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 greater 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.
- Chris 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 zonefile 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 (gpgsql) 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.

21.11.16 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 were 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 odbcbackend was removed. It only runs on Windows and Windows is unsupported since 3.0. Removal in commit 2576.
- We used to send the chunk length and the actual chunk in two separate writes (often resulting in two separate TCP packets) during outbound AXFR. This confused MSDNS. We now combine those writes. Code in commit 2575.
- The bindbackend can now run without SQLite3, as previously intended. Fix in commit 2574.
- Some high-concurrency master setups would crash under load. Fixed in commit 2571.

21.11.17 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 dnsdemog did not compile at all. Fixes in commit 2538, commit 2554.
- We now allow unescaped tabs in TXT records. Fix in commit 2539.
- SOA records no longer disappear during incoming transfers. Fix by Ruben d'Arco in commit 2540.
- PowerDNS compiles on OS X (and other platforms that support our auth server but not the recursor) again, fix in commit 2566.
- Cleanups related to warnings from gcc and valgrind in commit 2561, commit 2562, commit 2565.
- Solaris compatibility fixes by Ruben d'Arco, Juraj Lutter and others in commit 2548, commit 2552, commit 2553, commit 2560. Fixes for *BSD in commit 2546.
- pdns_control help would report 'version' twice, reported by Gerwin, fix in commit 2549.

DNSSEC related fixes

- When slaving zones, PowerDNS now automatically detects that a zone is 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 'ordername 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.
- GSQLite3 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 gpgsql schema is now constrained to lowercase, as PowerDNS would be unable to find other entries anyway. Fix in commit 2503, closing ticket 426.
- The gsql-backends can now handle huge records, thanks to a patch by Ruben d'Arco. Code in commit 2476, closing ticket 407. Additional changes in commit 2292, commit 2487, commit 2489. Closes ticket 218, ticket 316.
- Some of PowerDNS' internal classes would work with uninitialized data when repurposed outside of the PowerDNS core logic. Fix in commit 2469,
- pdnssec now has 'check-all-zones' and 'rectify-all-zones' commands. Submitted by Ruben d'Arco, code in commit 2467.
- 'restart' in our init.d-script would not start pdns if it was down before. Fixed in commit 2462.
- 'pdnssec rectify-zone' now honours ^^verbose and is rather quiet without it. Code in commit 2443.
- Improved error messages for systems without IPv6. Changes in commit 2425.
- The packet- and querycache now honour TTLs from backend data. Code in commit 2414.
- 'pdns_control help' now shows useful usage information. Code in commit 2410 and commit 2465.
- Jasper Spaans improved our init.d script for compliance with Debian Squeeze. Patch in commit 2251. Further improvement with 'set -e' to initscript contributed by Marc Haber in commit 2301.
- Klaus Darilion discovered our configuration file template and ^^help output explained the various cache TTLs wrongly, and he also added documentation for some missing parameters. commit 2271 and commit 2272.

- Add support for building against Botan 1.10 (stable) and drop support for 1.9 (development). Changes in commit 2334. This fixes several bugs when building against 1.9.
- Upgrade internal PolarSSL library to their version 1.1.1. Change in commit 2389 and beyond.
- Compilation of several backends failed for Boost in non-standard locations. Fixes in commit 2316..
- We now do additional processing for SRV records too. Code in commit 2388, closing ticket 423 (which also contained the patch). Regression test updates that flow from this in commit 2390.
- Fix compilation on OSX. commit 2316.
- Fix pdnssec crash when asked to do DNSSEC without a DNSSEC capable backend. Code in commit 2369.
- If PowerDNS was not configured to operate as a DNS master, it would still accept 'pdns_control notify' commands, but then not do it. Spotted by David Gavarret, patch by Jose Arthur Benetasso Villanova in commit 2379.
- In various places we would only accept UPPERCASE DNS typenames. Fixed in commit 2370, closing ticket 390.
- We would not always drop supplemental groups correctly. Reported by David Black of Atlassian.
- Our regression tests have been strengthened a lot, and now cover way more features. Commits in 2280, 2281, 2282, 2317, 2348, 2349, 2350, 2351 and beyond.
- Update to support the latest draft of DANE/TLSA. Spotted by James Cloos (commit 2338). Further improvements by Pieter Lexis in commit 2347, commit 2358.
- Compilation on OpenBSD was eased by patches from Brad Smith, which can be found in commit 2288 and commit 2291, closing ticket 95.
- 'make check' failed on the internal PolarSSL. Spotted by Daniel Briley, fix in commit 2283.
- The default SQL schemas were expanded to contain far longer content fields. commit 2292, commit 2293.
- Documentation typos, Jake Spencer (commit 2304), Jose Arthur Benetasso Villanova (commit 2337). Code typos in commit 2324 (closes ticket 296).
- Manpage updates from Debian, provided by Matthijs Möhlmann. Content in commit 2306.
- pdnssec rectify-zone can now accept multiple zones at the same time. Code in commit 2383.
- As suggested in ticket 416, we now log the SOA serial number after committing an AXFR'd 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"

21.11.18 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.

21.11.19 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)

21.11.20 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

21.11.21 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 PowerDNS's DNSSEC support 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.

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 Travaille (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, Vasiliy G Tolstov, Brielle Bruns, Evan Hunt, Ralf van der Enden, Marc Laros, Serge Belyshev, Chris 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 Chris 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-renotify' to retransmit notifies for slaved zones, which is helpful when acting as a 'signing slave' for a hidden master. Code in commit 1950.
- No longer let zone2sql and zone2ldap import BIND 'hint' zones. commit 1998.
- Allow for timestamps to explicitly be specified in (s)econds. Code in commit 1398, closing ticket 250.
- Zones with URL and MBOXFW records can be transferred over AXFR, code in commit 1464.
- Maik Zumstrull cleaned up the BIND Backend makefile, plus taught our init.d script to read /etc/default/pdns. Code in commit 1601, commit 1602.
- Generic SQL backends now support multiple masters in the domains table. Code in commit 1857. Additionally, masters can also have :port numbers. Code in commit 1858.

21.11.22 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 recvfrom errors, which used to spam the log on many systems. Addressed in commit 1320.
- Direct queries for 'fancy records' would lead to errors, such queries now fail early. Spotted by Jorn Ekkelenkamp, implemented in 1051.
- Fix typo in geobackend, closing ticket 157, implemented in 1090.
- Initial work on TSIG support - not done yet. Spurred on by Marco Davids.
- Embarrassingly, the 'master' configuration setting was not documented in the list of all settings!
- Norbert has updated OpenDBX so that SQLite reads and writes no longer deadlock, plus compilation fixes on Solaris, plus the addition of autoserials to backends that support triggers. Implemented in commit 1154.
- Random generator is now based on AES, improving the security of certain proxy operations. This is the same random generator that is in the recursor. Implemented in commit 1256.
- Documentation for 'supermaster' mode was improved due to popular demand.
- When binding to a UDP port failed, supply a more precise error message (commit 1245)
- The zone parser error messages were vastly improved, partially inspired by Shawn's cowboy migration. Code in commit 1235.
- Labels are compressed more efficiently (case-insensitively), leading to smaller packets. Implemented in commit 1156.
- Fix handling of TCP timeouts to not cause a reload of the backends. Implemented in commit 1092.
- TCP Receiver no longer spams the log with common network errors. Implemented in commit 1306.
- Move from select() to poll()-based multiplexing, allowing PowerDNS to listen on more than 1024 sockets simultaneously. One big PowerDNS user needs this. Implemented in 1072.
- Zone2sql now reads source files in performance enhancing inode order. Additionally, zone2sql no longer dies on a missing zone file if **^^on-error-resume-next** was specified. Finally, statistics of zone2sql conversion have been improved. Implemented in 1055.
- Address issues found by more recent g++ versions. Spotted and/or fixed by Jorn Ekkelenkamp (commit 1051), Marcus Rueckert (commit 1094), Norbert Sendetzky (commit 1107), Serge Belyshev (commit 1171).
- The Intel C Compiler implements certain things differently, causing the master/slave communicator to malfunction. Spotted by Marcus Rueckert, implemented in 1052, plus fallout in 1105.

- PowerDNS can now be compiled with Boost 1.37.0.
- Andre Lorbach of Adiscon discovered the Microsoft Windows 2003 nameserver adds out of zone data to zone transfers, which we need to ignore, instead of rejecting the entire zone. Implemented in 1048.
- PowerDNS now skips remote master servers which consistently generate timeout messages, improving the master checking cycle time tremendously. Developed in cooperation with Tyler Hall. Implemented in commit 1278.
- When binding to a UDP port failed, supply a more precise error message (commit 1245)
- **dnsreplay** now waits for the final answers to arrive, making it possible to process even small pcap files and get meaningful statistics. commit 1268.
- **dnsreplay** has a more sane default timeout now, which can be configured too. Suggested by Augie Schwer in ticket 163, implemented in commit 1287.

21.11.23 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.

21.11.24 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.

21.11.25 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.

21.11.26 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 benefiting 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.

21.11.27 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 is 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.

21.11.28 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.
- SSQLite 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 writev, 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.

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

21.11.30 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

21.11.31 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

21.11.32 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.

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

21.11.34 Version 2.9.12

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

New features

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

Bugs

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

Improvements, cleanups

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

21.11.35 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_recursor now cleans its cache, which should save some memory in the long run. Norbert contributed some small LDAP work which should come in useful in the future.

21.11.36 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_recursor may have logged empty lines for some users, fixed. Solution suggested by Norbert Sendetzky.
- LDAP: DNS TTLs were random values (Norbert Sendetzky, Stefan Pfetzing). New **ldap-default-ttl** option.
- LDAP: Now works with OpenLDAP 2.1 (Norbert Sendetzky)
- LDAP: error handling for invalid MX records implemented (Norbert Sendetzky)
- LDAP: better exception handling (Norbert Sendetzky)
- LDAP: code cleanup of lookup() (Norbert Sendetzky)
- LDAP: added support for scoped searches (Norbert Sendetzky)

21.11.37 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!

21.11.38 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 unparsable 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.

21.11.39 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.

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

- Brad Knowles noticed that zone2sql had problems with the root.zone, fixed. This also closes some other zone2sql annoyances with converting single zones.

21.11.41 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).

21.11.42 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
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.

21.11.43 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 one every database under the sun.

21.11.44 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 nonexistent 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).

21.11.45 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.

21.11.46 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.

21.11.47 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.

21.11.48 Version 2.6.1

Quick fix release for a big cache problem.

21.11.49 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 benchmarketing 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'.

21.11.50 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 ccounts** command outputs statistics on what is in the cache, which is useful to help optimize your caching strategy.

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

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

  The old Packet Cache is still there (and useful) but see 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 packetcache, such as when using recursion under high load.
- AXFR log messages now all contain the word 'AXFR' to ease grepping.
- Linux static version now compiled with gcc 3.2 which is known to output better and faster code than the previously used 3.0.4.

Bugs fixed

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

21.11.52 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.

21.11.53 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.

21.11.54 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*.

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

21.11.56 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.

21.11.57 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.

21.11.58 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.

21.11.59 Version 2.0 Release Candidate 1

The Mac OS 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.

21.11.60 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".

21.11.61 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)

21.11.62 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 Rijsdijk for suggesting saner defaults.
- Infrastructure for 'also-notify' statements added.

21.11.63 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.

21.11.64 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

21.11.65 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, oraclebackend

Some of these features will be present in newer releases.

21.11.66 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, oraclebackend

Some of these features will be present in newer releases.

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

Feature enhancements

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

Performance enhancements

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

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.

21.11.68 Version 1.99.4 Early Access Prerelease

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

Major bugs fixed

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

Feature enhancements

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

Performance enhancements

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

Known bugs

- FreeBSD version does not stop when requested to do so.
- Zone2sql refuses named.confs with less than 100 domains.
- Wildcard CNAMEs do not work as they do with bind.

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.

21.11.69 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.confs with large numbers of small domains, as well as small numbers of large domains (TLD).

Zone transfers are now also much improved.

Major bugs fixed - zone2sql leaked file descriptors on each domain, used wrong Bison recursion leading to parser stack overflows. This limited the amount of domains that could be parsed to 1024. - zone2sql can now read all known 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.

21.11.70 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.

21.11.71 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.

CHAPTER

TWENTYTWO

END OF LIFE STATEMENTS

We aim to have a release every six months. The latest release receives correctness, stability and security updates. The two releases before that get critical updates only. Older releases are marked end of life and receive no updates at all. Pre-releases do not receive immediate security updates.

The currently supported release train of PowerDNS Authoritative Server is 4.9.

PowerDNS Authoritative Server 4.8 will only receive critical updates; it will be end of life after PowerDNS Authoritative Server 5.1 is released.

PowerDNS Authoritative Server 4.7 will only receive critical updates; it will be end of life after PowerDNS Authoritative Server 5.0 is released.

PowerDNS Authoritative Server 4.0 through 4.6, 3.x, and 2.x are End of Life.

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.

Table 1: PowerDNS Authoritative Server Release Life Cycle

Version	Release date	Critical-Only updates	End of Life
4.9	15th of March 2024	~ March 2025	~ March 2026
4.8	1st of June 2023	15th of March 2024	~ September 2025
4.7	20th of October 2022	1st of June 2023	~ March 2025
4.6	25th of January 2022	20th of October 2022	EOL March 2024
4.5	July 13 2021	25th of January 2022	EOL June 2023
4.4	December 18 2020	25th of January 2022	EOL 20th of October 2022
4.3	April 7 2020	~ April 2021	EOL 25th of January 2022
4.2	August 30 2019	December 2020	EOL July 2021
4.1 and older	EOL	EOL	EOL

22.1 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 the latest version. 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.

22.2 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.x - it is a faster, more standards conforming and more powerful nameserver!

CHAPTER

TWENTYTHREE

FREQUENTLY ASKED QUESTIONS

This document lists categorized answers and questions with links to the relevant documentation.

23.1 Replication

Please note that not all PowerDNS Server backends support primary or secondary operation, see the *table of backends*.

23.1.1 My PowerDNS Authoritative Server does not send NOTIFY messages

Don't forget to enable primary support by setting *primary* to `yes` in your configuration. In *primary mode* PowerDNS Authoritative Server will send NOTIFYs to all nameservers that are listed as NS records in the zone by default.

23.1.2 My PowerDNS Authoritative Server does not start AXFRs

Don't forget to enable secondary-support by setting *secondary* to `yes` in your configuration. In *secondary mode* PowerDNS Authoritative Server listens for NOTIFYs from the primary IP for zones that are configured as secondary zones, and will also periodically check for SOA serial number changes at the primary.

23.1.3 Can PowerDNS Server act as Secondary and Primary at the same time?

Yes totally, enable both by saying `yes` to *primary* and *secondary* in your configuration.

23.1.4 How can I limit Zone Transfers (AXFR) per Domain?

With the ALLOW-AXFR-FROM metadata, See *the documentation*.

23.1.5 I have a working Autoprimary/Autosecondary setup but when I remove Zones from the Primary they still remain on the Secondary. 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 secondary manually or via a custom script.

23.2 Operational

23.2.1 The ADDITIONAL is section different than BIND's answer, why?

The PowerDNS Authoritative Server by default does not 'trust' other zones in its own database.

23.2.2 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!

23.2.3 Primary or Secondary support is not working, PowerDNS is not picking up changes

The Primary/Secondary apparatus is off by default. Turn it on by adding a *secondary* and/or *primary* statement to the configuration file. Also, check that the configured backend is primary or secondary capable and you entered exactly the same string to the Domains tables without the ending dot.

23.2.4 My primaries 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* setting.

23.2.5 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.

23.2.6 Linux Netfilter says your conntrack table is full?

Thats 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.

23.2.7 I get an error about writing to /etc

This may look something like "unable to open temporary zonefile '/etc/powerdns/zones/example.com.<random number>'". PowerDNS systemd units enable `ProtectSystem=full` by default, which disallows writes to `/etc` and `/usr`, among other places. Either move your zone files to a safer place (`/var/lib/powerdns` is a popular choice) or change the systemd protection settings.

For more background on this, please see the systemd documentation on ProtectSystem and ReadWritePaths.

23.3 Backends

23.3.1 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.

23.3.2 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.

23.3.3 Which backend should I use? There are so many!

If you have no external constraints, the *Generic MySQL/MariaDB backend*, *Generic PostgreSQL backend* and *Generic SQLite 3 backend* ones are probably the most used and complete.

The bindbackend is also pretty capable too in fact, but many prefer a relational database.

23.3.4 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.

23.3.5 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.

23.3.6 Can I specify custom sql queries for the gmysql / gpgsql backend or are those hardcoded?

Yes you can override the *default queries*.

CHAPTER

TWENTYFOUR

BACKEND WRITERS' GUIDE

PowerDNS backends are implemented via a simple yet powerful C++ interface. If your needs are not met by the regular backends, including the PipeBackend and the RemoteBackend, 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.

24.1 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.

24.2 Simple read-only native backends

Implementing a backend consists of inheriting from the DNSBackend class. For read-only backends, which do not support secondary operation, only the following methods are relevant:

```
class DNSBackend
{
public:

virtual void lookup(const QType &qtype, const string &qdomain, int zoneId=-1, DNSPacket *pkt_p=nullptr)=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);
};
```

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 can be found 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()`.

Figuring out the Start of Authority can require an important number of call to `getSOA()` if the name has a lot of labels. For example, figuring out that the SOA for `2.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.` is `d.0.1.0.0.2.ip6.arpa.` might involve 26 calls, chopping off one label at a time. If your backend has an efficient way to figure out the best SOA it has for a given name, it is possible to override the default `getSOA()` implementation to immediately return the `d.0.1.0.0.2.ip6.arpa.` SOA record to the first `2.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. getSOA()` call.

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 backends. 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.

Warning: Each zone should have a unique id, even across backends.

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;
```

24.3 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, int zoneId, DNSPacket *p)
  {
    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
{
```

(continues on next page)

(continued from previous page)

```
public:
  RandomFactory() : BackendFactory("random") {}

  DNSBackend *make(const string &suffix)
  {
    return new RandomBackend();
  }
};

/* THIRD PART */

class RandomLoader
{
public:
  RandomLoader()
  {
    BackendMakers().report(std::make_unique<RandomFactory>());
    g_log << 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 name, `random.powerdns.com`, another loaded backend would have to know about the SOA and NS records for the `powerdns.com` zone and such.

Warning: Spreading the content of a zone across multiple backends, described above as 'overlay', makes the zone incompatible with some operations that assume that a single zone is always entirely stored in the same backend. Such operations include zone transfers, listing and editing zone content via the API or `pdnsutil`.

Warning: When the content of a zone is spread across multiple backends, all the types for a given name should be delegated to the same backend. For example a backend can know about all the types for `random.powerdns.com` while another backend knows about all the types for `random2.powerdns.com`, but it is not possible to let one backend handle only `AAAA` queries for all names while another one handles only `A` queries, for example. This limitation comes from the fact that PowerDNS uses `ANY` queries to fetch all types from the backend in one go and that it assumes that once one backend has returned records the other ones do not need to be called. It is also possible to have two backends providing records for the same name and types, for example if the first one does not support DNSSEC and the second does, but that requires some mechanism outside of PowerDNS to keep records in sync between the two backends.

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.

Note: This simple backend neglects to handle case properly!

24.4 Interface definition

24.4.1 Classes

class DNSResourceRecord

DNSName *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 non-zero, 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

DNSName *SOAData*::**nameserver**
Name of the primary nameserver of this zone

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*::**minimum**
Minimum acceptable value for TTL

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 secondary

24.4.2 Methods

void `DNSBackend::`**lookup**(**const** QType *&qtype*, **const** string *&qdomain*, int *zoneId* = -1, DNSPacket **pkt* = nullptr)
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: Since 4.1.0, 'SOA' lookups are not passed this pointer anymore because PowerDNS doesn't support tailoring whether a whole zone exists or not based on who is asking.

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 PDNSException 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()` of `list()`. If this functions returns **true**, **rr** has been filled with data. When it returns false, no more data is available, and **rr** does not contain new data. A backend should make sure that it either fills out all fields of the DNSResourceRecord or resets them to their default values.

The qname field of the DNSResourceRecord should be filled out with the exact `qdomain` passed to lookup, preserving its case. So if a query for 'CaSe.yourdomain.com' comes in and your database contains data for 'case.yourdomain.com', the qname field of rr should contain 'CaSe.yourdomain.com'!

Should throw an PDNSException 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 true. If the backend is functioning correctly, but does not consider itself authoritative, it should return false. In case of errors, an PDNSException should be thrown.

24.5 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 `g_log`, in descending order of graveness.

24.6 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 `BackendFactory::`**`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 `BackendFactory::`**`declare`**`(`**`const`** string &*suffix*, **`const`** string &*param*, **`const`** string &*explanation*, **`const`** string &*value*`)`

The suffix is passed to your method, and can be passed on to declare. **param** is the name of your parameter. **explanation** is what will appear in the output of –help. Furthermore, a default value can be supplied in the **value** parameter.

A sample implementation:

```
void declareArguments(const string &suffix)
{
  declare(suffix,"dbname","Pdns backend database name to connect to","powerdns
↪");
  declare(suffix,"user","Pdns backend user to connect as","powerdns");
  declare(suffix,"host","Pdns backend host to connect to","");
  declare(suffix,"password","Pdns backend password to connect with","");
}
```

After the arguments have been declared, they can be accessed from your backend using the `mustDo()`, `getArg()` and `getArgAsNum()` methods. The are defined as follows in the DNSBackend class:

void `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 `strtol()` internally.

Sample usage from the BIND backend: 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;
}
```

24.7 Read/write secondary-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. Secondary domains are pulled from the primary. PowerDNS needs to know for which domains it is to be a secondary, and for each secondary domain, what the IP address of the primary is.

A secondary zone is pulled from a primary, 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 secondary nameserver needs to check at the primary if the serial number there is greater 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 `getUnfreshSecondaryInfos` method, which is called on all backends periodically. This method fills a vector of **DomainInfo**s 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, bool␣
↪getSerial = true);
       virtual bool isPrimary(const ComboAddress& ipAddress);
       virtual bool startTransaction(const string &qname, int id);
       virtual bool commitTransaction();
       virtual bool abortTransaction();
       virtual bool feedRecord(const DNSResourceRecord &rr, const DNSName &
↪ordername, bool ordernameIsNSEC3 = false);
       virtual void getUnfreshSecondaryInfos(vector<DomainInfo>* domains);
       virtual void setFresh(uint32_t id);
       /* ... */
}
```

The mentioned DomainInfo struct looks like this:

uint32_t `DomainInfo::`**`id`**
    ID of this zone within this backend

vector<ComboAddress> `DomainInfo::`**`primaries`**
    IP addresses of the primary of this domain (may be empty)

uint32_t `DomainInfo::`**`serial`**
    Serial number of this zone

uint32_t `DomainInfo::`**`notified_serial`**
    Last serial number of this zone that secondaries have seen

time_t `DomainInfo::`**`last_check`**
    Last time this zone was checked over at the primary 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 secondary

`enum DomainKind`
    The kind of domain, one of {Primary,Secondary,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 secondary 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::`**`isPrimary`**`(`**`const`** ComboAddress *&ipAddress*`)`

If a backend considers itself a secondary for the given domain and if the IP address in **ipAddress** is indeed a primary, it should return true. False otherwise. This is a first line of checks to guard against reloading a domain unnecessarily.

void `DomainInfo::`**`getUnfreshSecondaryInfos`**`(`vector<DomainInfo> *domains`)`

When called, the backend should examine its list of secondary domains and add any unfresh ones to the domains vector.

bool `DomainInfo::`**`getDomainInfo`**`(`**`const`** string *&name*, DomainInfo *&di*, boot *getSerial*`)`

This is like `getUnfreshSecondaryInfos`, 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*, **`const`** DNSName *&ordername*, bool *ordernameIsNSEC3*`)`

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 `getUnfreshSecondaryInfos()`.

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);
g_log<<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);
g_log<<Logger::Error<<"AXFR done for '"<<domain<<"'"<<endl;
```

24.8 Autoprimary/autosecondary capability

A backend that wants to act as an 'autosecondary' (formerly 'superslave') for a primary should implement the following method:

```
class DNSBackend
{
   virtual bool autoPrimaryBackend(const string &ip, const DNSName &domain, const␣
↪vector<DNSResourceRecord>&nsset, string *nameserver, string *account, DNSBackend␣
↪**db)
};
```

This function gets called with the IP address of the potential autoprimary, 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 autoprimary (if accounting is desired), and the db needs to be filled with a pointer to your backend.

Autoprimary/autosecondary is a complicated concept, if this is all unclear see the *Autoprimary: automatic provisioning of secondaries* documentation.

24.9 Read/write primary-capable backends

In order to be a useful primary 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 to secondary nameservers.

In order to do so, PowerDNS calls the `getUpdatedPrimaries()` method. Like the `getUnfreshSecondaryInfos()` 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 getUpdatedPrimaries(vector<DomainInfo>* domains, std::unordered_set
↪<DNSName> &catalogs, CatalogHashMap &catalogHashes);
  virtual void setNotified(uint32_t id, uint32_t serial);
      /* ... */
}
```

These functions all have a default implementation that doesn't do anything - which explains that these methods can be omitted in simple backends. Furthermore, unlike with simple backends, a secondary 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::`**`getUpdatedPrimaries`**(vector<DomainInfo> *domains, std::unordered_set<DNSName> &*catalogs*, CatalogHashMap &*catalogHashes*)

When called, the backend should examine its list of primary domains and add any changed ones to the `DomainInfo` vector.

void `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.

24.10 DNS update support

To make your backend DNS update compatible, it needs to implement a number of new functions and functions already used for secondary 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 DNSName &qname, int id);
  virtual bool commitTransaction();
  virtual bool abortTransaction();
  virtual bool feedRecord(const DNSResourceRecord &rr, DNSName &ordername, bool␣
↪ordernameIsNSEC3);
  virtual bool replaceRRSet(uint32_t domain_id, const DNSName& qname, const QType&␣
↪qt, const vector<DNSResourceRecord>& rrset)
  virtual bool listSubZone(const DNSName &zone, int domain_id);
  /* ... */
}
```

virtual bool DNSBackend::**startTransaction**(**const** DNSName *&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*, **const** DNSName *&ordername*, bool *ordernameIsNSEC3*)

See *above*. Please keep in mind that the zone is not empty because `startTransaction()` was called differently.

virtual bool DNSBackend::**listSubZone**(**const** DNSName *&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** DNSName *&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.

24.11 Domain metadata support

As described in *Per zone settings: Domain Metadata*, each served zone can have "metadata". Such metadata determines how this zone behaves in certain circumstances. In order for a backend to support domain metadata, the following operations have to be implemented:

```
class DNSBackend {
public:
  /* ... */
  virtual bool getAllDomainMetadata(const DNSName& name, std::map<std::string,␣
↪std::vector<std::string> >& meta);
```

(continues on next page)

(continued from previous page)

```
  virtual bool getDomainMetadata(const DNSName& name, const std::string& kind,␣
↪std::vector<std::string>& meta);
  virtual bool setDomainMetadata(const DNSName& name, const std::string& kind,␣
↪const std::vector<std::string>& meta);
  /* ... */
}
```

virtual bool **getAllDomainMetadata**(**const** DNSName *&name*, std::map<std::string, std::vector<std::string>> *&meta*)

Fills 'meta' with the value(s) of all kinds for zone 'name'. Returns true if the domain metadata operation are supported, regardless of whether there is any data for this zone.

virtual bool **getDomainMetadata**(**const** DNSName *&name*, **const** std::string *&kind*, std::vector<std::string> *&meta*)

Fills 'meta' with the value(s) of the specified kind for zone 'name'. Returns true if the domain metadata operation are supported, regardless of whether there is any data of this kind for this zone.

virtual bool **setDomainMetadata**(**const** DNSName *&name*, **const** std::string *&kind*, **const** std::vector<std::string> *&meta*)

Store the values from 'meta' for the specified kind for zone 'name', discarding existing values if any. An empty meta is equivalent to a deletion request. Returns true if the values have been correctly stored, and false otherwise.

24.12 TSIG keys

In order for a backend to support the storage of TSIG keys, the following operations have to be implemented:

```
class DNSBackend {
public:
  /* ... */
  virtual bool getTSIGKey(const DNSName& name, DNSName& algorithm, string&␣
↪content);
  virtual bool setTSIGKey(const DNSName& name, const DNSName& algorithm, const␣
↪string& content);
  virtual bool deleteTSIGKey(const DNSName& name);
  virtual bool getTSIGKeys(std::vector< struct TSIGKey > &keys);
  /* ... */
}
```

24.13 DNSSEC support

In order for a backend to support DNSSEC, quite a few number of additional operations have to be implemented:

```
struct KeyData {
  std::string content;
  unsigned int id;
  unsigned int flags;
  bool active;
  bool published;
};

class DNSBackend {
public:
  /* ... */
  virtual bool doesDNSSEC();
  virtual bool getBeforeAndAfterNamesAbsolute(uint32_t id, const DNSName& qname,␣
↪DNSName& unhashed, DNSName& before, DNSName& after);
```

(continues on next page)

(continued from previous page)

```
  /* update operations */
  virtual bool updateDNSSECOrderNameAndAuth(uint32_t domain_id, const DNSName& qname, const DNSName& ordername, bool auth, const uint16_t qtype=QType::ANY);
  virtual bool updateEmptyNonTerminals(uint32_t domain_id, set<DNSName>& insert, set<DNSName>& erase, bool remove);
  virtual bool feedEnts(int domain_id, map<DNSName,bool> &nonterm);
  virtual bool feedEnts3(int domain_id, const DNSName &domain, map<DNSName,bool> &nonterm, const NSEC3PARAMRecordContent& ns3prc, bool narrow);

  /* keys management */
  virtual bool getDomainKeys(const DNSName& name, std::vector<KeyData>& keys);
  virtual bool removeDomainKey(const DNSName& name, unsigned int id);
  virtual bool addDomainKey(const DNSName& name, const KeyData& key, int64_t& id);
  virtual bool activateDomainKey(const DNSName& name, unsigned int id);
  virtual bool deactivateDomainKey(const DNSName& name, unsigned int id);
  virtual bool publishDomainKey(const DNSName& name, unsigned int id);
  virtual bool unpublishDomainKey(const DNSName& name, unsigned int id);

  /* ... */
}
```

`virtual` bool **`doesDNSSEC`**()

Returns true if that backend supports DNSSEC.

`virtual` bool **`getBeforeAndAfterNamesAbsolute`**(uint32_t *id*, `const` DNSName *&qname*, DNSName *&unhashed*, DNSName *&before*, DNSName *&after*)

Asks the names before and after qname for NSEC and NSEC3. The qname will be hashed when using NSEC3. Care must be taken to handle wrap-around when qname is the first or last in the ordered list of zone names. Please note that in case the requested name is present in the zone, it should be returned as the "before" name.

`virtual` bool **`updateDNSSECOrderNameAndAuth`**(uint32_t *domain_id*, `const` DNSName *&qname*, `const` DNSName *&ordername*, bool *auth*, `const` uint16_t *qtype* = QType::ANY)

Updates the ordername and auth fields.

`virtual` bool **`updateEmptyNonTerminals`**(uint32_t *domain_id*, set<DNSName> *&insert*, set<DNSName> *&erase*, bool *remove*)

Updates ENT after a zone has been rectified. If 'remove' is false, 'erase' contains a list of ENTs to remove from the zone before adding any. Otherwise all ENTs should be removed from the zone before adding any. 'insert' contains the list of ENTs to add to the zone after the removals have been done.

`virtual` bool **`feedEnts`**(int *domain_id*, map<DNSName, bool> *&nonterm*)

This method is used by `pdnsutil rectify-zone` to populate missing non-terminals. This is used when you have, say, record like _sip._upd.example.com, but no _udp.example.com. PowerDNS requires that there exists a non-terminal in between, and this instructs you to add one.

`virtual` bool **`feedEnts3`**(int *domain_id*, `const` DNSName *&domain*, map<DNSName, bool> *&nonterm*, `const` NSEC3PARAMRecordContent *&ns3prc*, bool *narrow*)

Same as feedEnts, but provides NSEC3 hashing parameters.

`virtual` bool **`getDomainKeys`**(`const` DNSName *&name*, std::vector<KeyData> *&keys*)

Retrieves all DNSSEC keys. Content must be valid key record in format that PowerDNS understands.

`virtual` bool **`removeDomainKey`**(`const` DNSName *&name*, unsigned int *id*)

Removes this key.

`virtual` bool **`addDomainKey`**(`const` DNSName *&name*, `const` KeyData *&key*, int64_t *&id*)

Adds a new DNSSEC key for this domain.

virtual bool **activateDomainKey**(**const** DNSName &*name*, unsigned int *id*)
Activates an inactive DNSSEC key for this domain.

virtual bool **deactivateDomainKey**(**const** DNSName &*name*, unsigned int *id*)
Deactivates an active DNSSEC key for this domain.

virtual bool **publishDomainKey**(**const** DNSName &*name*, unsigned int *id*)
Publishes a previously hidden DNSSEC key for this domain.

virtual bool **unpublishDomainKey**(**const** DNSName &*name*, unsigned int *id*)
Hides a DNSSEC key for this domain. Hidden DNSSEC keys are used for signing but do not appear in the actual zone, and are useful for rollover operations.

24.14 Miscellaneous

24.14.1 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)

24.14.2 Storage classes

You may have noticed that PowerDNS has several C++ classes for holding DNS data. Some use presentation format, some use the wire format. Some just hold content, some hold a whole record.

Below, we'll show the class definitions of each (with some details omitted, but with some useful words added) to help you find your way.

```
struct DNSZoneRecord
{
  int domain_id{-1};
  uint8_t scopeMask{0};
  int signttl{0};
  DNSName wildcardname;
  bool auth{true};
  bool disabled{false};
  DNSRecord dr;
};
```

`DNSZoneRecord` holds a record in the context of a zone. It is a wrapper around `DNSRecord` with some extra fields that PowerDNS might need to handle DNSSEC and ECS correctly.

```
struct DNSRecord
{
  DNSRecord() : d_type(0), d_class(QClass::IN), d_ttl(0), d_clen(0), d_
↪place(DNSResourceRecord::ANSWER)
  {}
  explicit DNSRecord(const DNSResourceRecord& rr);
  DNSName d_name;
  std::shared_ptr<DNSRecordContent> d_content;
  uint16_t d_type;
  uint16_t d_class;
  uint32_t d_ttl;
  uint16_t d_clen;
  DNSResourceRecord::Place d_place;

  // this orders by name/type/class/ttl/lowercased zone representation
  bool operator<(const DNSRecord& rhs);
```

(continues on next page)

(continued from previous page)

```
  // this orders in canonical order and keeps the SOA record on top
  static bool prettyCompare(const DNSRecord& a, const DNSRecord& b);

  bool operator==(const DNSRecord& rhs) const
};
```

DNSRecord holds a DNS record. It has name, type, class, TTL, content length, and a content object of type DNSRecordContent.

```
class DNSRecordContent
{
public:
  static std::shared_ptr<DNSRecordContent> make(...);

  virtual std::string getZoneRepresentation(bool noDot=false) const = 0;
  virtual void toPacket(DNSPacketWriter& pw) const =0;
  string serialize(const DNSName& qname, bool canonic=false, bool lowerCase=false);
  virtual bool operator==(const DNSRecordContent& rhs); // compares presentation format
  static shared_ptr<DNSRecordContent> deserialize(const DNSName& qname, uint16_t qtype, const string& serialized, uint16_t qclass=QClass::IN);

  void doRecordCheck(const struct DNSRecord&){}

  virtual uint16_t getType() const = 0;
};
```

DNSRecordContent holds DNS content, in individual fields for the various contents of record types. It is subclassed for all supported types:

```
class SRVRecordContent : public DNSRecordContent
{
public:
  SRVRecordContent(uint16_t preference, uint16_t weight, uint16_t port, DNSName target);

  includeboilerplate(SRV)

  uint16_t d_weight, d_port;
  DNSName d_target;
  uint16_t d_preference;
};
```

```
class DNSResourceRecord
{
public:
  DNSResourceRecord() : last_modified(0), ttl(0), signttl(0), domain_id(-1), qclass(1), scopeMask(0), auth(true), disabled(false) {};
  static DNSResourceRecord fromWire(const DNSRecord& d);

  void setContent(const string& content);
  string getZoneRepresentation(bool noDot=false) const;

  DNSName qname; //!< the name of this record, for example: www.powerdns.com
  DNSName ordername;
  DNSName wildcardname;
  string content; //!< what this record points to. Example: 10.1.2.3

  uint32_t ttl; //!< Time To Live of this record
  uint32_t signttl; //!< If non-zero, use this TTL as original TTL in the RRSIG
```

(continues on next page)

(continued from previous page)

```
  int domain_id; //!< If a backend implements this, the domain_id of the zone this record is in
  QType qtype; //!< qtype of this record, ie A, CNAME, MX etc
  uint16_t qclass; //!< class of this record

  uint8_t scopeMask;
  bool auth;
  bool disabled;

  bool operator==(const DNSResourceRecord& rhs);

  bool operator<(const DNSResourceRecord &b);
```

`DNSResourceRecord` holds a DNS record with content in presentation format, as a string.

CHAPTER

TWENTYFIVE

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* modules are compiled into the binary. The *pipe* is, by default, compiled as a runtime loadable module.

25.1 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 `autoreconf -vi` in the clone.

You can also download snapshot tarballs.

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
- D630 0CAB CBF4 69BB E392 E503 A208 ED4F 8AF5 8446
- 16E1 2866 B773 8C73 976A 5743 6FFC 3343 9B0D 04DF
- 990C 3D0E AC7C 275D C6B1 8436 EACA B90B 1963 EC2B

There is a PGP keyblock with these keys available on https://doc.powerdns.com/powerdns-keyblock.asc.

Older releases (4.3.x and earlier) can also be signed with one of the following keys:

- 1628 90D0 689D D12D D33E 4696 1C5E E990 D2E7 1575
- B76C D467 1C09 68BA A87D E61C 5E50 715B F2FF E1A7

25.2 Dependencies

To build the PowerDNS Authoritative Server, a C++ compiler with support for C++ 2017 is required. This means gcc 7.1 and newer and clang 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 1.54 or newer
- OpenSSL

To build from a Git repository clone, the following dependencies are also required:

- ragel
- bison
- flex
- Python 3.6 or newer, with the 'venv' package

25.3 Optional dependencies

Several options that can be passed to `./configure` can enable and disable different features. These will require additional dependencies

25.3.1 ed25519 support with libsodium

The PowerDNS Authoritative Server can link with libsodium to support ed25519 (DNSSEC algorithm 15). To detect libsodium, use the `--with-libsodium` configure option.

25.3.2 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`.

CHAPTER

TWENTYSIX

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. Starting with 4.2.0, linking in Botan is no longer possible.

Optionally, PowerDNS can link in a copy of the open source Sodium library.

26.1 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 *https://downloads.powerdns.com/releases*.

If absolute certainty is required, we recommend consulting an expert in US Export Control, or asking the BIS for confirmation.

CHAPTER

TWENTYSEVEN

INTERNALS

27.1 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.

CHAPTER

TWENTYEIGHT

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.

28.1 A

The A record contains an IP address. It is stored as a decimal dotted quad string, for example: '203.0.113.210'.

28.2 AAAA

The AAAA record contains an IPv6 address. An example: '2001:DB8:2000:bf0::1'.

28.3 AFSDB

A specialised record type for the 'Andrew Filesystem'. Stored as: '#subtype hostname', where subtype is a number.

28.4 ALIAS

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.

28.5 APL

The APL record, specified in **RFC 3123**, is used to specify a DNS RR type "APL" for address prefix lists.

28.6 CAA

The “Certification Authority Authorization” record, specified in **RFC 6844**, is used to specify Certificate Authorities that may issue certificates for a domain.

28.7 CERT

Specialised record type for storing certificates, defined in **RFC 2538**.

28.8 CDNSKEY

The CDNSKEY (**Child DNSKEY**) type is supported.

28.9 CDS

The CDS (**Child DS**) type is supported.

28.10 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’.

28.11 CSYNC

The CSYNC record is used for ‘Child-to-Parent Synchronization in DNS’, as described in **RFC 7477**. Right now it is only supported as zone content; no special processing is implemented. Note that SOA-EDIT is not applied to serial numbers in CSYNC content.

28.12 DNSKEY

The DNSKEY DNSSEC record type is fully supported, as described in **RFC 4034**. Enabling DNSSEC for domains can be done with *pdnsutil*.

28.13 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.

28.14 DS

The DS DNSSEC record type is fully supported, as described in **RFC 4034**. Enabling DNSSEC for domains can be done with *pdnsutil*.

28.15 HINFO

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

28.16 HTTPS

See *SVCB* for more information.

28.17 KEY

The KEY record is fully supported. For its syntax, see **RFC 2535**.

28.18 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`

28.19 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’.

28.20 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.

28.21 NS

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

28.22 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*.

28.23 OPENPGPKEY

The OPENPGPKEY records, specified in **RFC 7929**, are used to bind OpenPGP certificates to email addresses.

28.24 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.

28.25 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.

28.26 RRSIG

The RRSIG DNSSEC record type is fully supported, as described in **RFC 4034**.

28.27 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 minimum
```

Besides the primary and the hostmaster, all fields are numerical. The fields have complicated and sometimes controversial meanings.

28.28 SPF

SPF records can be used to store Sender Policy Framework details (**RFC 4408**).

28.29 SSHFP

The SSHFP record type, used for storing Secure Shell (SSH) fingerprints, is fully supported. A sample from **RFC 4255** is:

```
2 1 123456789abcdef67890123456789abcdef67890
```

28.30 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.

28.31 SVCB, HTTPS

New in version 4.4.0.

SVCB records, defined in (draft-ietf-dnsop-svcb-https-07) are used to facilitate the lookup of information needed to make connections to network services. SVCB records allow a service to be provided from multiple alternative endpoints, each with associated parameters (such as transport protocol configuration and keys for encrypting the TLS ClientHello). They also enable aliasing of apex domains, which is not possible with CNAME. The HTTPS RR is a variation of SVCB for HTTPS and HTTP origins.

Additional processing is supported for these types. Some *PowerDNS extensions* for automatic IP address hints exist as well.

28.32 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.

28.33 TLSA

Since 3.0. The TLSA records, specified in **RFC 6698**, are used to bind SSL/TLS certificate to named hosts and ports.

28.34 SMIMEA

Since 4.1. The SMIMEA record type, specified in **RFC 8162**, is used to bind S/MIME certificates to domains.

28.35 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.

28.36 URI

The URI record, specified in **RFC 7553**, is used to publish mappings from hostnames to URIs.

28.37 ZONEMD

The ZONEMD record, specified in **RFC 8976**, is used to validate zones.

28.38 Other types

The following, rarely used or obsolete record types, are also supported:

- DHCID (**RFC 4701**)
- DLV (**RFC 4431**)
- EUI48/EUI64 (**RFC 7043**)
- IPSECKEY (**RFC 4025**)
- KEY (**RFC 2535**, obsolete)
- KX (**RFC 2230**)
- L32 (**RFC 6742**)
- L64 (**RFC 6742**)
- LP (**RFC 6742**)
- MINFO (**RFC 1035**)
- MR (**RFC 1035**)
- NID (**RFC 6742**)
- RKEY (draft-reid-dnsext-rkey-00.txt)

28.39 Unknown DNS Resource Record (RR) Types

PowerDNS supports (**RFC 3597**) syntax for serving unknown record types. For example

```
e.example.   IN          TYPE1               \# 4 0A000001
```

Beware that PowerDNS will attempt to parse known record types even if written in this syntax. This bug will be fixed in future release.

CHAPTER

TWENTYNINE

POWERDNS/DNSDIST LICENSE

We remind PowerDNS and dnsdist users that under the terms of the GNU General Public License, PowerDNS and dnsdist come with ABSOLUTELY NO WARRANTY.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Lesser General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
```

(continues on next page)

(continued from previous page)

```
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
```

(continues on next page)

(continued from previous page)

```
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
```

(continues on next page)

(continued from previous page)

```
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
```

(continues on next page)

(continued from previous page)

```
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

                            NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
```

(continues on next page)

(continued from previous page)

```
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.
```

HTTP ROUTING TABLE

INDEX