
Loop User Manual

Release 1.99.7.20250831123527.772ce56e35

Banu Systems Private Limited

Aug 31, 2025

CONTENTS

1	Introduction	3
1.1	Scope of Document	3
1.2	Organization of This Document	3
1.3	Conventions Used in This Document	3
1.4	The Domain Name System (DNS)	4
1.4.1	DNS Fundamentals	4
1.4.2	Domains and Domain Names	4
1.4.3	Zones	5
1.4.4	Authoritative Name Servers	5
1.4.4.1	The Primary Master	6
1.4.4.2	Slave Servers	6
1.4.4.3	Stealth Servers	6
1.4.5	Caching Name Servers	7
1.4.5.1	Forwarding	7
1.4.6	Name Servers in Multiple Roles	7
2	Installation	9
2.1	Hardware requirements	9
2.2	Supported platforms	10
2.2.1	Problems with SELinux	11
2.3	Upgrading	12
3	Operation	13
3.1	Sample Configurations	13
3.1.1	A Caching-only Name Server	13
3.1.2	An Authoritative-only Name Server	13
3.2	Load-balancing	14
3.3	Name Server Operations	15
3.3.1	Tools for Use With the Name Server Daemon	15
3.3.1.1	Diagnostic Tools	15
3.3.1.2	Administrative Tools	16
4	Programs	19
4.1	named --- DNS nameserver that implements authoritative server and recursive resolver features	19
4.1.1	Synopsis	19

4.1.2	Description	19
4.1.3	Options	19
4.1.4	Signals	22
4.1.5	Files	23
4.1.6	See also	23
4.1.7	Copyright	23
4.2	named-journalprint --- Zone journal file printer	23
4.2.1	Synopsis	23
4.2.2	Description	23
4.2.3	See also	23
4.2.4	Copyright	24
4.3	rndc --- Remote-control a <i>named(8)</i> process	24
4.3.1	Synopsis	24
4.3.2	Description	24
4.3.3	Options	24
4.3.4	Commands	25
4.3.5	Files	30
4.3.6	See also	30
4.3.7	Copyright	30
4.4	nsec3hash --- NSEC3 hash generator	31
4.4.1	Synopsis	31
4.4.2	Description	31
4.4.3	Options	31
4.4.4	Copyright	31
4.5	arpaname --- IP address to ARPA name translator	31
4.5.1	Synopsis	31
4.5.2	Description	31
4.5.3	See also	32
4.5.4	Copyright	32
4.6	rndc-confgen --- <i>rndc(1)</i> key generator	32
4.6.1	Synopsis	32
4.6.2	Description	32
4.6.3	Options	32
4.6.4	Examples	33
4.6.5	See also	33
4.6.6	Copyright	34
4.7	ddns-confgen --- TSIG key generator	34
4.7.1	Synopsis	34
4.7.2	Description	34
4.7.3	Options	34
4.7.4	See also	35
4.7.5	Copyright	35
4.8	named-checkconf --- <i>named.conf</i> syntax checker	35
4.8.1	Synopsis	35
4.8.2	Description	35
4.8.3	Options	36
4.8.4	Exit status	36
4.8.5	See also	36

4.8.6	Copyright	36
4.9	named-checkzone --- Zone master file syntax checker	37
4.9.1	Synopsis	37
4.9.2	Description	37
4.9.3	Options	37
4.9.4	Exit status	39
4.9.5	See also	39
4.9.6	Copyright	39
4.10	named-rrchecker --- Resource records syntax checker	40
4.10.1	Synopsis	40
4.10.2	Description	40
4.10.3	Options	40
4.10.4	Examples	40
4.10.5	See also	41
4.10.6	Copyright	41
4.11	dnssec-dsfromkey --- DS RR generator	41
4.11.1	Synopsis	41
4.11.2	Description	41
4.11.3	Options	41
4.11.4	Examples	42
4.11.5	Files	42
4.11.6	Caveat	43
4.11.7	See also	43
4.11.8	Copyright	43
4.12	dnssec-importkey --- DNSKEY importer	43
4.12.1	Synopsis	43
4.12.2	Description	43
4.12.3	Options	43
4.12.4	Timing options	44
4.12.5	Files	44
4.12.6	See also	44
4.12.7	Copyright	44
4.13	dnssec-keyfromlabel --- DNSKEY importer from HSM	45
4.13.1	Synopsis	45
4.13.2	Description	45
4.13.3	Options	45
4.13.4	Timing options	47
4.13.5	Generated key files	48
4.13.6	See also	49
4.13.7	Copyright	49
4.14	dnssec-keygen --- DNSKEY and KEY generator	49
4.14.1	Synopsis	49
4.14.2	Description	49
4.14.3	Options	49
4.14.4	Timing options	52
4.14.5	Generated key files	53
4.14.6	Examples	54
4.14.7	See also	54

4.14.8	Copyright	54
4.15	dnssec-revoke --- DNSKEY revoker	54
4.15.1	Synopsis	54
4.15.2	Description	55
4.15.3	Options	55
4.15.4	See also	55
4.15.5	Copyright	55
4.16	dnssec-settime --- DNSKEY timing metadata setter	55
4.16.1	Synopsis	55
4.16.2	Description	56
4.16.3	Options	56
4.16.4	Timing options	57
4.16.5	Printing options	58
4.16.6	See also	58
4.16.7	Copyright	58
4.17	dnssec-signzone --- DNSSEC zone signer	58
4.17.1	Synopsis	58
4.17.2	Description	58
4.17.3	Options	59
4.17.4	Examples	64
4.17.5	See also	64
4.17.6	Copyright	64
4.18	dnssec-verify --- DNSSEC zone signature verifier	65
4.18.1	Synopsis	65
4.18.2	Description	65
4.18.3	Options	65
4.18.4	See also	66
4.18.5	Copyright	66
4.19	delv --- DNSSEC lookup and validation utility	66
4.19.1	Synopsis	66
4.19.2	Description	66
4.19.3	Options	67
4.19.4	Query options	69
4.19.5	Files	71
4.19.6	See also	71
4.19.7	Copyright	71
4.20	dig --- DNS client with comprehensive DNS query capabilities	72
4.20.1	Synopsis	72
4.20.2	Description	72
4.20.3	Options	72
4.20.4	Query options	75
4.20.5	Multiple queries	82
4.20.6	Files	82
4.20.7	See also	83
4.20.8	Copyright	83
4.21	host - Simple DNS lookup client	83
4.21.1	Synopsis	83
4.21.2	Description	83

4.21.3	Options	83
4.21.4	Files	86
4.21.5	See also	86
4.21.6	Copyright	86
4.22	nsupdate - DNS client to send dynamic DNS UPDATES to a nameserver	86
4.22.1	Synopsis	86
4.22.2	Description	86
4.22.3	Options	87
4.22.4	Input commands	89
4.22.5	Examples	91
4.22.6	Files	92
4.22.7	See also	92
4.22.8	Copyright	92
4.23	dnssperf --- DNS client that measures DNS nameserver performance	92
4.23.1	Synopsis	92
4.23.2	Description	92
4.23.2.1	Configuring the name server	93
4.23.2.2	Constructing a query input file	93
4.23.2.3	Constructing a dynamic update input file	93
4.23.2.4	Running the tests	94
4.23.3	Options	94
4.23.4	See also	96
4.23.5	Copyright	96
4.24	resperf --- DNS client that measures DNS resolver performance	96
4.24.1	Synopsis	96
4.24.2	Description	97
4.24.2.1	How resperf works	97
4.24.2.2	What you will need	98
4.24.2.3	Running the test	99
4.24.2.4	Interpreting the report	100
4.24.2.5	Determining the server's maximum throughput	101
4.24.2.6	Generating constant traffic	102
4.24.3	The plot data file	102
4.24.4	Options	103
4.24.5	See also	105
4.24.6	Copyright	105
5	Configuration	107
5.1	named.conf --- named program's configuration	107
5.1.1	Description	107
5.1.2	Configuration grammar	107
5.1.2.1	Data types	108
5.1.2.2	acl statement	110
5.1.2.3	controls statement	110
5.1.2.4	include statement	112
5.1.2.5	key statement	112
5.1.2.6	logging statement	113
5.1.2.6.1	channel phrase	114

5.1.2.6.2	category phrase	117
5.1.2.7	managed-keys statement	122
5.1.2.8	masters statement	123
5.1.2.9	options statement	124
5.1.2.10	server statement	165
5.1.2.11	trusted-keys statement	169
5.1.2.12	view statement	169
5.1.2.13	zone statement	177
5.1.2.13.1	Zone Types	179
5.1.2.13.2	Class	181
5.1.2.13.3	Zone Options	181
5.1.3	Comments syntax	185
5.1.4	Files	186
5.1.5	See also	187
5.1.6	Copyright	187
5.2	<code>rndc.conf</code> --- rndc program's configuration	187
5.2.1	Description	187
5.2.2	Configuration grammar	187
5.2.2.1	key statement	187
5.2.2.2	server statement	188
5.2.2.3	options statement	188
5.2.3	Comments syntax	189
5.2.4	Examples	190
5.2.5	Nameserver configuration	191
5.2.6	See also	191
5.2.7	Copyright	191
6	Features	193
6.1	Resolver	193
6.1.1	Smoothed round-trip time computation	193
6.1.2	Nameserver selection	194
6.2	DNS NOTIFY	194
6.3	DNS UPDATE (dynamic updates)	195
6.3.1	Journal files	195
6.4	IXFR (incremental zone transfers)	196
6.5	Split DNS	196
6.5.1	Example split DNS setup	197
6.5.2	Example split DNS setup 2	200
6.6	TSIG (transaction signatures)	201
6.6.1	Generating a shared key	201
6.6.2	Loading a new key	202
6.6.3	Instructing the server to use a key	202
6.6.4	TSIG-based access control	203
6.6.5	TSIG errors	203
6.7	TKEY (transaction keys)	204
6.8	SIG(0)	204
6.9	DNSSEC	205
6.9.1	Generating Keys	205

6.9.2	Signing the Zone	206
6.9.3	Configuring Servers	206
6.10	DNSSEC, Dynamic Zones, and Automatic Signing	208
6.10.1	Converting from insecure to secure	208
6.10.2	Dynamic DNS update method	209
6.10.3	Fully automatic zone signing	210
6.10.4	Private-type records	211
6.10.5	DNSKEY rollovers	211
6.10.6	Dynamic DNS update method	211
6.10.7	Automatic key rollovers	212
6.10.8	NSEC3PARAM rollovers via UPDATE	212
6.10.9	Converting from NSEC to NSEC3	212
6.10.10	Converting from NSEC3 to NSEC	212
6.10.11	Converting from secure to insecure	212
6.10.12	Periodic re-signing	212
6.10.13	NSEC3 and OPTOUT	213
6.11	Dynamic Trust Anchor Management	213
6.11.1	Validating Resolver	213
6.11.2	Authoritative Server	213
6.12	PKCS#11	214
6.12.1	Example of PKCS#11 usage	214
6.12.1.1	Install dependencies	215
6.12.1.2	Create a working directory	215
6.12.1.3	Configure SoftHSMv2	215
6.12.1.4	Configure OpenSSL	217
6.12.1.5	Create a DNSKEY by generating or importing keys	218
6.12.1.5.1	Create a DNSKEY by generating it	218
6.12.1.5.2	Create a DNSKEY by importing it from the HSM	220
6.12.1.6	Sign a DNS zone using the DNSKEY	221
6.12.1.7	Verify the signed zone's data	225
6.13	IPv6	225
6.13.1	Address Lookups Using AAAA Records	225
6.13.2	Address to Name Lookups Using Nibble Format	226
6.14	Empty zones	226
6.15	DNS64	230
6.16	Forwarding queries	230
6.17	Dual-stack Servers	230
6.18	Access Control	230
6.19	Interfaces	231
6.20	Query address	232
6.21	Zone Transfers	233
6.22	UDP port lists	234
6.23	Operating System Resource Limits	235
6.24	Server Resource Limits	235
6.25	Periodic Task Intervals	235
6.26	RRset ordering	236
6.27	Tuning	236
6.28	Built-in server information zones	236

6.29	Built-in Empty Zones	237
6.30	Content Filtering	237
6.31	Response Policy Zones (RPZ)	238
6.32	Response Rate Limiting (RRL)	243
6.32.1	Dynamic Update Policies	246
6.32.2	Multiple views	249
7	Master Files	251
7.1	Types of Resource Records and When to Use Them	251
7.1.1	Resource Records	251
7.1.2	Textual expression of RRs	254
7.2	Discussion of MX Records	255
7.3	Setting TTLs	255
7.4	Inverse Mapping in IPv4	256
7.5	Directives	256
7.5.1	@	256
7.5.2	\$ORIGIN	257
7.5.3	\$INCLUDE	257
7.5.4	\$TTL	257
7.5.5	\$GENERATE	257
8	Statistics	261
8.1	XML Format	263
8.2	Counters	263
8.3	Nameserver Counters	263
8.4	Zone Maintenance Counters	265
8.5	Resolver Counters	265
8.6	Socket I/O Counters	266
9	Security considerations	269
9.1	Access Control Lists	269
9.2	chroot() and setuid()	270
9.2.1	The chroot Environment	270
9.2.2	Using the setuid Function	270
9.3	DNS UPDATE security	271
9.4	Control channel security	271
10	Troubleshooting	273
10.1	Common Problems	273
10.1.1	It's not working; how can I figure out what's wrong?	273
10.2	Incrementing and Changing the Serial Number	273
11	Release notes	275
11.1	Loop 1.99.7	275
11.2	Loop 1.99.6	277
11.3	Loop 1.99.5	277
11.4	Loop 1.99.4	278
11.5	Loop 1.99.3	280
11.6	Loop version numbering scheme	281

11.6.1	Stable and development versions	282
11.7	Loop branches	282
11.8	History of Loop	282
12	License	285
13	Data and privacy	297
	Index	299

Warning

This document is still a work-in-progress. Large parts of it may be inaccurate, and it may mention programs, features and configuration options that do not exist in Loop. This warning will be removed when the manual is known to be correct.

INTRODUCTION

The Internet Domain Name System (DNS) consists of the syntax to specify the names of entities in the Internet in a hierarchical manner, the rules used for delegating authority over names, and the system implementation that actually maps names to Internet addresses. DNS data is maintained in a group of distributed hierarchical databases.

1.1 Scope of Document

Loop implements a domain name server for a number of operating systems. This document provides basic information about the installation and care of the Loop software package for system administrators.

1.2 Organization of This Document

In this document, *Chapter 1* introduces the basic DNS and Loop concepts. *Chapter 2* describes resource requirements for running Loop in various environments. Information in *Chapter 3* is *task-oriented* in its presentation and is organized functionally, to aid in the process of installing the Loop software. The task-oriented section is followed by *Chapter 4*, which contains more advanced concepts that the system administrator may need for implementing certain options. The contents of *Chapter 6* are organized as in a reference manual to aid in the ongoing maintenance of the software. *Chapter 7* addresses security considerations, and *Chapter 8* contains troubleshooting help. The main body of the document is followed by several *appendices* which contain useful reference information, such as a *bibliography* and historic information related to the Domain Name System.

1.3 Conventions Used in This Document

In this document, we use the following general typographic conventions:

<i>To describe:</i>	<i>We use the style:</i>
a pathname, filename, URL, hostname, mailing list name, or new term or concept	Fixed width
literal user input	Fixed Width Bold
program output	Fixed Width

The following conventions are used in descriptions of the Loop configuration file:

<i>To describe:</i>	<i>We use the style:</i>
keywords	Fixed Width
variables	Fixed Width
Optional input	[Text is enclosed in square brackets]

1.4 The Domain Name System (DNS)

The purpose of this document is to explain the installation and upkeep of the Loop software package, and we begin by reviewing the fundamentals of the Domain Name System (DNS) as they relate to Loop.

1.4.1 DNS Fundamentals

The Domain Name System (DNS) is a hierarchical, distributed database. It stores information for mapping Internet host names to IP addresses and vice versa, mail routing information, and other data used by Internet applications.

Clients look up information in the DNS by calling a *resolver* library, which sends queries to one or more *name servers* and interprets the responses. The Loop software distribution contains a name server called `named`.

1.4.2 Domains and Domain Names

The data stored in the DNS is identified by *domain names* that are organized as a tree according to organizational or administrative boundaries. Each node of the tree, called a *domain*, is given a label. The domain name of the node is the concatenation of all the labels on the path from the node to the *root* node. This is represented in written form as a string of labels listed from right to left and separated by dots. A label need only be unique within its parent domain.

For example, a domain name for a host at the company *Example, Inc.* could be `ourhost.example.com`, where `com` is the top level domain to which `ourhost.example.com` belongs, `example` is a subdomain of `com`, and `ourhost` is the name of the host.

For administrative purposes, the name space is partitioned into areas called *zones*, each starting at a node and extending down to the leaf nodes or to nodes where other zones

start. The data for each zone is stored in a *name server*, which answers queries about the zone using the *DNS protocol*.

The data associated with each domain name is stored in the form of *resource records* (RRs). Some of the supported resource record types are described in *section_title*.

For more detailed information about the design of the DNS and the DNS protocol, please refer to the standards documents listed in *section_title*.

1.4.3 Zones

To properly operate a name server, it is important to understand the difference between a *zone* and a *domain*.

As stated previously, a zone is a point of delegation in the DNS tree. A zone consists of those contiguous parts of the domain tree for which a name server has complete information and over which it has authority. It contains all domain names from a certain point downward in the domain tree except those which are delegated to other zones. A delegation point is marked by one or more *NS records* in the parent zone, which should be matched by equivalent NS records at the root of the delegated zone.

For instance, consider the `example.com` domain which includes names such as `host.aaa.example.com` and `host.bbb.example.com` even though the `example.com` zone includes only delegations for the `aaa.example.com` and `bbb.example.com` zones. A zone can map exactly to a single domain, but could also include only part of a domain, the rest of which could be delegated to other name servers. Every name in the DNS tree is a *domain*, even if it is *terminal*, that is, has no *subdomains*. Every subdomain is a domain and every domain except the root is also a subdomain. The terminology is not intuitive and we suggest that you read RFCs 1033, 1034 and 1035 to gain a complete understanding of this difficult and subtle topic.

Though Loop is called a "domain name server", it deals primarily in terms of zones. The master and slave declarations in the `named.conf` file specify zones, not domains. When you ask some other site if it is willing to be a slave server for your *domain*, you are actually asking for slave service for some collection of zones.

1.4.4 Authoritative Name Servers

Each zone is served by at least one *authoritative name server*, which contains the complete data for the zone. To make the DNS tolerant of server and network failures, most zones have two or more authoritative servers, on different networks.

Responses from authoritative servers have the "authoritative answer" (AA) bit set in the response packets. This makes them easy to identify when debugging DNS configurations using tools like `dig` (*section_title*).

1.4.4.1 The Primary Master

The authoritative server where the master copy of the zone data is maintained is called the *primary master* server, or simply the *primary*. Typically it loads the zone contents from some local file edited by humans or perhaps generated mechanically from some other local file which is edited by humans. This file is called the *zone file* or *master file*.

In some cases, however, the master file may not be edited by humans at all, but may instead be the result of *dynamic update* operations.

1.4.4.2 Slave Servers

The other authoritative servers, the *slave* servers (also known as *secondary* servers) load the zone contents from another server using a replication process known as a *zone transfer*. Typically the data are transferred directly from the primary master, but it is also possible to transfer it from another slave. In other words, a slave server may itself act as a master to a subordinate slave server.

Periodically, the slave server must send a refresh query to determine whether the zone contents have been updated. This is done by sending a query for the zone's SOA record and checking whether the SERIAL field has been updated; if so, a new transfer request is initiated. The timing of these refresh queries is controlled by the SOA REFRESH and RETRY fields, but can be overridden with the `max-refresh-time`, `min-refresh-time`, `max-retry-time`, and `min-retry-time` options.

If the zone data cannot be updated within the time specified by the SOA EXPIRE option (up to a hard-coded maximum of 24 weeks) then the slave zone expires and will no longer respond to queries.

1.4.4.3 Stealth Servers

Usually all of the zone's authoritative servers are listed in NS records in the parent zone. These NS records constitute a *delegation* of the zone from the parent. The authoritative servers are also listed in the zone file itself, at the *top level* or *apex* of the zone. You can list servers in the zone's top-level NS records that are not in the parent's NS delegation, but you cannot list servers in the parent's delegation that are not present at the zone's top level.

A *stealth server* is a server that is authoritative for a zone but is not listed in that zone's NS records. Stealth servers can be used for keeping a local copy of a zone to speed up access to the zone's records or to make sure that the zone is available even if all the "official" servers for the zone are inaccessible.

A configuration where the primary master server itself is a stealth server is often referred to as a "hidden primary" configuration. One use for this configuration is when the primary master is behind a firewall and therefore unable to communicate directly with the outside world.

1.4.5 Caching Name Servers

The resolver libraries provided by most operating systems are *stub resolvers*, meaning that they are not capable of performing the full DNS resolution process by themselves by talking directly to the authoritative servers. Instead, they rely on a local name server to perform the resolution on their behalf. Such a server is called a *recursive* name server; it performs *recursive lookups* for local clients.

To improve performance, recursive servers cache the results of the lookups they perform. Since the processes of recursion and caching are intimately connected, the terms *recursive server* and *caching server* are often used synonymously.

The length of time for which a record may be retained in the cache of a caching name server is controlled by the Time To Live (TTL) field associated with each resource record.

1.4.5.1 Forwarding

Even a caching name server does not necessarily perform the complete recursive lookup itself. Instead, it can *forward* some or all of the queries that it cannot satisfy from its cache to another caching name server, commonly referred to as a *forwarder*.

There may be one or more forwarders, and they are queried in turn until the list is exhausted or an answer is found. Forwarders are typically used when you do not wish all the servers at a given site to interact directly with the rest of the Internet servers. A typical scenario would involve a number of internal DNS servers and an Internet firewall. Servers unable to pass packets through the firewall would forward to the server that can do it, and that server would query the Internet DNS servers on the internal server's behalf.

1.4.6 Name Servers in Multiple Roles

The Loop name server can simultaneously act as a master for some zones, a slave for other zones, and as a caching (recursive) server for a set of local clients.

However, since the functions of authoritative name service and caching/recursive name service are logically separate, it is often advantageous to run them on separate server machines. A server that only provides authoritative name service (an *authoritative-only* server) can run with recursion disabled, improving reliability and security. A server that is not authoritative for any zones and only provides recursive service to local clients (a *caching-only* server) does not need to be reachable from the Internet at large and can be placed inside a firewall.

INSTALLATION

2.1 Hardware requirements

Loop is packaged for a set of supported operating system platforms (see [Supported platforms](#)). Any machine (real or virtual) with one of these platforms can be used to run Loop.

DNS hardware requirements have traditionally been quite modest. For many installations, servers that have been pensioned off from active duty can perform admirably as DNS servers. For serving a handful of static zones with low traffic, even low-performance machines may be sufficient. If the server's operational duties are larger, then a suitably performant machine can be selected.

Loop's nameserver is multi-threaded, allowing utilization of multiprocessor systems for installations that need it.

The memory of the server has to be large enough to fit the cache and zones loaded off disk. The **max-cache-size** option of *named.conf(5)* can be used to limit the amount of memory used by the cache, at the expense of reducing cache hit rates and causing more DNS traffic. It is still good practice to have enough memory to load all zone and cache data into memory --- the best way to determine this for a given installation is to watch the nameserver in operation. After a few weeks the nameserver process should reach a relatively stable size where entries are expiring from the cache as fast as they are being inserted.

We aren't able to recommend specifications in this document as it would be outdated quickly. It is best to profile the usage patterns and prepare a hardware configuration accordingly.

Error

TODO: Add a link to Loop support for help with hardware configuration.

Error

TODO: Add a link to a tuning section.

2.2 Supported platforms

Loop is written to run on POSIX operating systems. The following platforms are supported by this release of Loop:

- Red Hat Enterprise Linux 8 (x86_64)
- Red Hat Enterprise Linux 8 (aarch64)
- Red Hat Enterprise Linux 9 (x86_64)
- Red Hat Enterprise Linux 9 (aarch64)
- Red Hat Enterprise Linux 10 (x86_64)
- Red Hat Enterprise Linux 10 (aarch64)
- Fedora 41 (x86_64)
- Fedora 41 (aarch64)
- Fedora 42 (x86_64)
- Fedora 42 (aarch64)

Users of AlmaLinux, Rocky Linux, and Oracle Linux distributions may use the packages for the corresponding Red Hat Enterprise Linux version. Packages for current versions of FreeBSD, Debian, and Ubuntu will be added in the future.

Installation instructions for each supported platform will be available soon. For now, if you know how to install RPMs using **dnf**, please adapt the following instructions for your platform.

To install Loop on Red Hat Enterprise Linux 10 (x86_64), you may run the following commands as the `root` user.

First, install the `akira-release` RPM package that will add the `akira-epel` and `akira-epel-testing` DNF repositories to your system, as well as associated GPG keys used to verify signed RPM packages from these repositories:

```
# dnf install https://download.banuu.com/packages/akira/1.99/
↪epel/testing/10/x86_64/akira-release-1.99.7.20250831123527.
↪772ce56e35-1.el10.noarch.rpm
```

Then, enable the `akira-epel-testing` DNF repository:

```
# dnf config-manager setopt akira-epel-testing.enabled=1
```

Then, install the `loop` RPM package that will install the Loop software and documentation:

```
# dnf install loop
```

Then, if you wish to run the nameserver, configure **named** suitably by editing `/etc/loop/named.conf`, and then run it:

```
# systemctl enable --now named
```

Note

For information about Loop's version numbering, see [Loop version numbering scheme](#).
For information about Loop's branches and EOL dates, see [Loop branches](#).

2.2.1 Problems with SELinux

On some distributions that have SELinux enabled, you may notice errors when running the **named** service such as:

```
Aug 10 07:59:06 rpi3 audit[14591]: AVC avc: denied { create_
↳ } for pid=14591 comm="loop-worker-0" name="tmp-dW3tOeMfdD"
↳ scontext=system_u:system_r:named_t:s0 tcontext=system_
↳ u:object_r:var_lib_t:s0 tclass=file permissive=0
Aug 10 07:59:06 rpi3 audit[14591]: AVC avc: denied { read_
↳ write open } for pid=14591 comm="loop-worker-0" path="/var/
↳ lib/loop/tmp-dW3tOeMfdD" dev="mmcblk0p3" ino=258270
↳ scontext=system_u:system_r:named_t:s0 tcontext=system_
↳ u:object_r:var_lib_t:s0 tclass=file permissive=0
Aug 10 07:59:07 rpi3 audit[14591]: AVC avc: denied { rename_
↳ } for pid=14591 comm="loop-worker-0" name="tmp-dW3tOeMfdD"
↳ dev="mmcblk0p3" ino=258270 scontext=system_u:system_r:named_
↳ t:s0 tcontext=system_u:object_r:var_lib_t:s0 tclass=file
↳ permissive=0
Aug 10 07:59:07 rpi3 audit[14591]: AVC avc: denied { unlink_
↳ } for pid=14591 comm="loop-worker-0" name="managed-keys.
↳ loop" dev="mmcblk0p3" ino=258227 scontext=system_u:system_
↳ r:named_t:s0 tcontext=system_u:object_r:var_lib_t:s0
↳ tclass=file permissive=0
```

These errors occur because SELinux, when using the *targeted* policy, runs the program with path `/usr/sbin/named` confined in the *named_t* security context. It limits the directories where the **named** process can write to. This can be verified by running the command:

```
$ ps axZ | grep named
system_u:system_r:named_t:s0      14591 ?          Ssl    0:01 /
↳ usr/sbin/named -u loop
```

The Loop package doesn't (and shouldn't) do anything to solve this issue automatically as it would be a hack. The SELinux policies for programs such as `/usr/sbin/named` are installed by a different package called *selinux-policy-targeted*, and are not handled by the Loop package.

You can workaround this issue by either configuring SELinux to run in *permissive*

mode, or by editing the *targeted* policy to remove the Loop programs from it.

We will rename **named** to **loopd** in a future build upon which this issue should not occur anymore.

2.3 Upgrading

Please read the release notes before upgrading to a newer version of Loop (see the chapter titled *Release notes*).

OPERATION

In this chapter we provide some suggested configurations along with guidelines for their use. We suggest reasonable values for certain option settings.

3.1 Sample Configurations

3.1.1 A Caching-only Name Server

The following sample configuration is appropriate for a caching-only name server for use by clients internal to a corporation. All queries from outside clients are refused using the `allow-query` option. Alternatively, the same effect could be achieved using suitable firewall rules.

```
// Two corporate subnets we wish to allow queries from.
acl corpnets { 192.168.4.0/24; 192.168.7.0/24; };
options {
    // Working directory
    directory "/etc/namedb";

    allow-query { corpnets; };
};
// Provide a reverse mapping for the loopback
// address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
    type master;
    file "localhost.rev";
    notify no;
};
```

3.1.2 An Authoritative-only Name Server

This sample configuration is for an authoritative-only server that is the master server for "example.com" and a slave for the subdomain "eng.example.com".

```
options {
    // Working directory
```

(continues on next page)

(continued from previous page)

```
directory "/etc/namedb";
// Do not allow access to cache
allow-query-cache { none; };
// This is the default
allow-query { any; };
// Do not provide recursive service
recursion no;
};

// Provide a reverse mapping for the loopback
// address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
    type master;
    file "localhost.rev";
    notify no;
};
// We are the master server for example.com
zone "example.com" {
    type master;
    file "example.com.db";
    // IP addresses of slave servers allowed to
    // transfer example.com
    allow-transfer {
        192.168.4.14;
        192.168.5.53;
    };
};
// We are a slave server for eng.example.com
zone "eng.example.com" {
    type slave;
    file "eng.example.com.bk";
    // IP address of eng.example.com master server
    masters { 192.168.4.12; };
};
```

3.2 Load-balancing

A primitive form of load-balancing can be achieved in the DNS by using multiple records (such as multiple A records) for one name. For example, if you have three webserver with network addresses of 10.0.0.1, 10.0.0.2, and 10.0.0.3, a set of records such as the following means that clients will connect to each machine one third of the time:

Name	TTL	RRCLASS	RRTYPE	Resource Record Data (RDATA)
www	600	IN	A	10.0.0.1
www	600	IN	A	10.0.0.2
www	600	IN	A	10.0.0.3

When a resolver queries for these records, Loop will rotate them and respond to the query with the records in a different order. In the example above, clients will randomly receive records in the order 1, 2, 3; 2, 3, 1; and 3, 1, 2. Most clients will use the first record returned and discard the rest.

For more detail on ordering responses, see [RRset ordering](#).

3.3 Name Server Operations

3.3.1 Tools for Use With the Name Server Daemon

This section describes several indispensable diagnostic, administrative and monitoring tools available to the system administrator for controlling and debugging the name server daemon.

3.3.1.1 Diagnostic Tools

The `dig` and `host` programs are command line tools for manually querying name servers. They differ in style and output format.

dig

`dig` is the most versatile and complete of these lookup tools. It has two modes: simple interactive mode for a single query, and batch mode which executes a query for each in a list of several query lines. All query options are accessible from the command line.

`dig @server domain query-type query-class + query-option - dig-option % comment` The usual simple use of `dig` will take the form

```
dig @server domain query-type query-class
```

For more information and a list of available commands and options, see the `dig` man page.

host

The `host` utility emphasizes simplicity and ease of use. By default, it converts between host names and Internet addresses, but its functionality can be extended with the use of options.

`host -aCdlnrsTwv -c class -N ndots -t type -W timeout -R retries -m flag -4 -6 hostname server` For more information and a list of available commands and options, see the `host` man page.

3.3.1.2 Administrative Tools

Administrative tools play an integral part in the management of a server.

named-checkconf

The `named-checkconf` program checks the syntax of a `named.conf` file.

`named-checkconf -jvz -t directory filename`

named-checkzone

The `named-checkzone` program checks a master file for syntax and consistency.

`named-checkzone -djvD -c class -o output -t directory -w directory -k (ignore | warn | fail) -n (ignore | warn | fail) -W (ignore | warn) zone filename`

rndc

The remote name daemon control (`rndc`) program allows the system administrator to control the operation of a name server. If you run `rndc` without any options it will display a usage message as follows:

`rndc -c config -s server -p port -y key command command` See ??? for details of the available `rndc` commands.

`rndc` requires a configuration file, since all communication with the server is authenticated with digital signatures that rely on a shared secret, and there is no way to provide that secret other than with a configuration file. The default location for the `rndc` configuration file is `/etc/loop/rndc.conf`, but an alternate location can be specified with the `-c` option. If the configuration file is not found, `rndc` will also look in `/etc/loop/rndc.key`. The `rndc.key` file is generated by running `rndc-confgen -a` as described in *section_title*.

The format of the configuration file is similar to that of `named.conf`, but limited to only four statements, the `options`, `key`, `server` and `include` statements. These statements are what associate the secret keys to the servers with which they are meant to be shared. The order of statements is not significant.

The `options` statement has three clauses: `default-server`, `default-key`, and `default-port`. `default-server` takes a host name or address argument and represents the server that will be contacted if no `-s` option is provided on the command line. `default-key` takes the name of a key as its argument, as defined by a `key` statement. `default-port` specifies the port to which `rndc` should connect if no port is given on the command line or in a `server` statement.

The `key` statement defines a key to be used by `rndc` when authenticating with `named`. Its syntax is identical to the `key` statement in `named.conf`. The keyword `key` is followed by a key name, which must be a valid domain name, though it need not actually be hierarchical; thus, a string like `"rndc_key"` is a valid name. The `key` statement has two clauses: `algorithm` and `secret`. While the configuration parser will accept any string as the argument to `algorithm`, currently only the strings `"hmac-md5"`, `"hmac-sha1"`, `"hmac-sha224"`, `"hmac-sha256"`, `"hmac-sha384"` and `"hmac-sha512"` have any meaning. The `secret` is a Base64 encoded string as specified in RFC 3548.

The `server` statement associates a key defined using the `key` statement with a server. The keyword `server` is followed by a host name or address. The `server` statement has two clauses: `key` and `port`. The `key` clause specifies the name of the key to be used when communicating with this server, and the `port` clause can be used to specify the port `rndc` should connect to on the server.

A sample minimal configuration file is as follows:

```
key rndc_key {
    algorithm "hmac-sha256";
    secret
    ↪ "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K
    ↪ ";
};
options {
    default-server 127.0.0.1;
    default-key    rndc_key;
};
```

This file, if installed as `/etc/loop/rndc.conf`, would allow the command:

```
$rndc reload
```

to connect to 127.0.0.1 port 953 and cause the name server to reload, if a name server on the local machine were running with following controls statements:

```
controls {
    inet 127.0.0.1
        allow { localhost; } keys { rndc_key; };
};
```

and it had an identical key statement for `rndc_key`.

Running the `rndc-confgen` program will conveniently create a `rndc.conf` file for you, and also display the corresponding `controls` statement that you need to add to `named.conf`. Alternatively, you can run `rndc-confgen -a` to set up a `rndc.key` file and not modify `named.conf` at all.

PROGRAMS

The following sections document programs that are part of the Loop software distribution.

4.1 **named** --- DNS nameserver that implements authoritative server and recursive resolver features

4.1.1 Synopsis

```
named [[-4] | [-6]] [-c <config-file>] [-d <debug-level>] [-D <id-string>] [-f] [-F] [-g] [-n  
<thread-count>] [-p <port>] [-s] [-S <maximum-sockets>] [-t <directory>] [-T <option>] [-u  
<username>] [-U <listener-count>] [-x <cache-file>]  
  
named [-h | -V]
```

4.1.2 Description

named is a general-purpose DNS nameserver daemon. It can be used as an authoritative server to serve zones, or as a caching recursive resolver, or in mixed mode. It implements a variety of features for use by DNS operators. See the Loop User Manual for detailed documentation on using it.

When invoked without arguments, **named** will read its default configuration file `/etc/loop/named.conf`, read any initial data, and listen for queries. A complete description of **named**'s configuration file is provided in *named.conf(5)* and in the Loop User Manual.

named inherits the umask (file creation mode mask) from its parent process. If files created by **named** (such as journal files) need to have custom permissions, the umask should be set explicitly in the environment used to start the **named** process.

4.1.3 Options

-4

Use IPv4 only even if the host machine is capable of IPv6. `-4` and `-6` are mutually exclusive.

-6

Use IPv6 only even if the host machine is capable of IPv4. **-4** and **-6** are mutually exclusive.

-c <config-file>

Use <config-file> as the **named** configuration file. The default is `/etc/loop/named.conf`.

Note

To ensure that reloading the configuration file continues to work after the server has changed its working directory due to a possible *directory* option in the configuration file, <config-file> should be an absolute pathname.

-d <debug-level>

Set the daemon's debug level to *debug-level*. Log messages from **named** become more verbose as the debug level increases. The default value is 0.

-D <id-string>

Specifies a string that may be used to identify a **named** process in a process listing. The contents of *id-string* are not examined or used. There is no default value.

-f

Don't daemonize the **named** process. Run it in the foreground.

-F

This is a reserved option.

Warning

This option is mainly of interest to Loop developers and may be removed or changed in a future release. Currently, **named** will not start if passed this argument.

-g

Don't daemonize the **named** process. Run it in the foreground. Force all logging to standard-error.

-h

Print program usage information and exit.

-n <thread-count>

Create <thread-count> worker threads to take advantage of multiple processors. If not specified, **named** will try to determine the number of processors present in the system and create one thread per processor. If it is unable to determine the number of processors, a single thread will be created.

-p <port>

Listen for queries on the specified <port> number. The default is 53.

-s

Write memory usage statistics to standard-output on exit.

Warning

This option is mainly of interest to Loop developers and may be removed or changed in a future release.

-S <maximum-sockets>

Allow **named** to use up to <maximum-sockets> sockets. The default value is 21000.

Warning

This option should be unnecessary for the vast majority of users. The use of this option could even be harmful because the specified value may exceed the limitation of the underlying system API. It is meant to be set only when the default configuration causes exhaustion of file descriptors and the operational environment is known to support the specified number of sockets. Note also that the actual maximum number is normally a little lower than the specified value because **named** reserves some file descriptors for its internal use.

-t <directory>

chroot (2) to <directory> after processing the command line arguments, but before reading the configuration file.

Note

This option should be used in conjunction with the *-u* option, as chrooting a process running as root doesn't enhance security on most systems; the way *chroot* (2) is defined allows a process with root privileges to escape a chroot jail.

-T <option>

This argument is used to pass undocumented <option>s used in testing. It is not meant for use by an operator.

Warning

This option is present for testing only. It is only of interest to Loop developers and may be removed or changed in a future release. Please don't use it.

-u <username>

setuid(2) to user <username> after completing privileged operations, such as creating sockets that listen on privileged ports.

On Linux, **named** uses the kernel's capability mechanism to drop all root privileges except the ability to *bind(2)* to a privileged port (`CAP_NET_BIND_SERVICE`) and set process resource limits (`CAP_SYS_RESOURCE`).

-U <listener-count>

Use <listener-count> threads to listen for incoming UDP datagrams on each address. If not specified, **named** will try to determine the number of processors present in the system and create one thread per processor. If it is unable to determine the number of processors, a single thread will be created. If *-n* has been set to a higher value than the number of detected processors, then *-U* may be increased as high as that value, but no higher.

-v

Print program version and exit.

-x <cache-file>

Load data from <cache-file> into the cache of the default view.

Warning

This option is present for testing only. It is only of interest to Loop developers and may be removed or changed in a future release. Please don't use it.

4.1.4 Signals

Certain UNIX signals cause the nameserver to take specific actions:

SIGHUP

Causes the nameserver to read its configuration file, and reload its zones.

SIGINT

Shuts down the nameserver.

SIGTERM

Shuts down the nameserver.

Signals can be sent using the *kill(1)* program. The result of sending any other signals to the server is undefined.

Note

In routine operation, signals should not be used to control **named**; **rndc** commands such as `reload` and `stop` should be used instead to instruct **named** to perform an action.

4.1.5 Files

`/etc/loop/named.conf`

The default configuration file. A complete description of **named**'s configuration file is provided in *named.conf* (5) and in the Loop User Manual.

`/run/loop/named.pid`

The default process ID file.

4.1.6 See also

named-checkconf (1), *named-checkzone* (1), *rndc* (1), *named.conf* (5)

4.1.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2001, 2003-2009, 2011, 2013-2018 Internet Systems Consortium, Inc. ("ISC").

4.2 **named-journalprint** --- Zone journal file printer

4.2.1 Synopsis

named-journalprint <*journal-file*>

4.2.2 Description

named-journalprint prints the contents of a zone journal file in a human-readable form.

Journal files are automatically created by *named* (8) when changes are made to dynamic zones (e.g., by DNS UPDATE or IXFR zone transfers). They record each addition or deletion of a resource record, in binary format, allowing the changes to be re-applied to the zone when the server is restarted after a shutdown or crash. By default, the name of the journal file is formed by appending the extension *.jnl* to the name of the corresponding zone file.

named-journalprint converts the contents of a given journal file into a human-readable text format. Each line begins with "add" or "del", to indicate whether the record was added or deleted, and continues with the resource record in master-file format.

4.2.3 See also

named (8), *nsupdate* (1)

4.2.4 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.3 rndc --- Remote-control a *named*(8) process

4.3.1 Synopsis

```
rndc [-b <source-address>] [-c <config-file>] [-k <key-file>] [-s <server>] [-p <port>] [-q]
[-v] [-y <key_id>] <command>
```

```
rndc [-h | -V]
```

4.3.2 Description

rndc is a remote-control program that can be used to send control commands to a running *named*(8) process. If **rndc** is invoked with no arguments, it prints a short summary of the supported commands and their arguments.

rndc communicates with the *named*(8) process over a control channel that uses a TCP connection, on which it sends commands authenticated using HMACs ([RFC 2104](#)). A common shared secret is used by both peers to authenticate control channel messages. All commands sent over the control channel must be signed by a <key_id> known to the server.

rndc reads a configuration file to determine how to contact a specific *named*(8) process, and to decide what HMAC algorithm and key it should use.

4.3.3 Options

-b <source-address>

Use <source-address> as the source address for the connection to the server. Multiple instances are permitted to allow setting of both the IPv4 and IPv6 source addresses.

-c <config-file>

Use <config-file> as the **rndc** configuration file. The default is /etc/loop/rndc.conf.

-h

Print program usage information and exit.

-k <key-file>

The key in <key-file> will be used to authenticate commands sent to the server if the config-file does not exist. The default is /etc/loop/rndc.key.

-s <server>

<server> is the name or address of the server which matches a server statement in the configuration file for **rndc**. If no **-s** option is supplied, by default **rndc**

uses the host named by the **default-server** clause in the **options** statement of the **rndc** configuration file.

-p <port>

Send control channel commands to the specified <port>. The default is 953.

-q

Enable quiet mode, where message text returned by the server will not be printed except when there is an error.

-v

Enable verbose logging.

-V

Print program version and exit.

-y <key_id>

Use the key identified by <key_id> from the **rndc** configuration file. <key_id> must be known by *named(8)* with the same algorithm and secret string in order for control channel message validation to succeed. If no **-y** option is supplied, by default **rndc** looks for a **key** clause in the **server** statement of the server being used, and if no **server** statement is present for that host, then the **default-key** clause of the **options** statement.

Warning

The **rndc** configuration file contains shared secrets which are used to send authenticated control channel messages to *named(8)* processes. It should therefore not have general read or write access.

<command>

<command> is the command that should be sent to the nameserver. See the next section for a list of commands.

4.3.4 Commands

A list of commands supported by **rndc** can be seen by running **rndc** without arguments. Currently supported commands are:

addzone <zone> [<class> [<view>]] <configuration>

Add a zone while the server is running. This command requires the **allow-new-zones** configuration option of *named.conf(5)* to be set to *yes*. The configuration string specified on the command line is the zone configuration text that would ordinarily be placed in *named.conf(5)*.

The configuration is saved in a file named <hash>.nzf, where <hash> is a cryptographic hash generated from the name of the view. When *named(8)* is restarted, the file will be loaded into the view configuration, so that zones that were added can persist after a restart.

The following example would add the zone *example.com* to the default view:

```
$ rndc addzone example.com '{ type master; file "example.  
↳com.db"; };'
```

(Note the curly braces and semi-colons in the zone configuration.)

Also see the *delzone* command.

delzone [-clean] <zone> [<class> [<view>]]

Delete a zone while the server is running. Only zones that were originally added using the *addzone* command can be deleted in this manner.

If the *-clean* argument is specified, the zone's master file (and journal file, if any) will be deleted along with the zone. Without the *-clean* option, zone files must be cleaned up by hand. (If the zone is of type *slave* or *stub*, the files needing to be cleaned up will be reported in the output of the *delzone* command.)

Also see the *addzone* command.

dumpdb (-all | -cache | -zones | -adb | -bad) [<view> ...]

Dump the server's caches (default) and/or zones to the dump file for the specified views. If no view was specified, all views are dumped.

Also see the *dump-file* configuration option of *named.conf(5)*.

flush

Flush the nameserver's resolver cache.

flushname <name> [<view>]

Flush the given <name> from the nameserver's resolver cache and, if applicable, from the nameserver's address database (ADB) and the bad-server cache.

flushtree <name> [<view>]

Flushes the given <name> and all of its subdomains from the nameserver's resolver cache, the address database (ADB), and the bad-server cache.

freeze [<zone> [<class> [<view>]]]

Suspend updates to a dynamic zone. If no <zone> is specified, then all zones are suspended. This allows manual edits to be made to a zone normally updated by dynamic update. It also causes changes in the journal file to be synced into the master file. All dynamic update attempts will be refused while the zone is frozen.

Also see the *thaw* command.

halt [-p]

Stop the server immediately. Recent changes made through dynamic update or IXFR are not saved to the master files, but will be rolled forward from the journal files when the server is restarted. If **-p** is specified, the *named(8)* process's PID is returned. This allows an external process to determine when *named(8)* had completed halting.

Also see the *stop* command.

loadkeys <zone> [<class> [<view>]]

Fetch all DNSSEC keys for the given <zone> from the key directory. If they are within their publication period, merge them into the zone's DNSKEY RRset. Unlike the *sign* command however, the zone is not immediately re-signed by the new keys, but is allowed to incrementally re-sign over time.

This command requires that the **auto-dnssec** zone configuration option of *named.conf(5)* be set to *maintain*, and also requires the zone to be configured to allow dynamic updates. (See "Dynamic Update Policies" in the Loop User Manual for more details.)

Also see the *sign* command.

notify <zone> [<class> [<view>]]

Resend NOTIFY messages for the zone.

notrace

Sets the server's debugging level to 0.

Also see the *trace* command.

querylog (on | off)

Enable or disable query logging. This command can also be used without an argument to toggle query logging on and off.

Query logging can also be configured by explicitly directing the *queries* log category to a *channel* in the *logging* section of the *named.conf(5)* configuration, or by using the **querylog** configuration option of *named.conf(5)*.

reconfig

Reload the configuration file and load new zones, but do not reload existing zone files even if they have changed. This is faster than a full *reload* command when there is a large number of zones because it avoids the need to examine the modification times of the zones files.

recurring

Dump the list of queries *named(8)* is currently recursing on, and the list of domains to which iterative queries are currently being sent. The second list includes the number of fetches currently active for the given domain, and how many have been passed or dropped because of the **fetches-per-zone** configuration option of *named.conf(5)*.

refresh <zone> [<class> [<view>]]

Schedule zone maintenance for the given <zone>.

reload [<zone> [<class> [<view>]]]

Reload the given <zone>. If <zone> is not specified, it reloads the configuration file of **named**, and its zones.

retransfer <zone> [<class> [<view>]]

Retransfer the given slave <zone> from the master server.

If the `<zone>` is configured to use **inline-signing** configuration option of `named.conf(5)`, the signed version of the zone is discarded; after the retransfer of the unsigned version is complete, the signed version will be regenerated with all new signatures.

scan

Scan the list of available network interfaces for changes, without executing a full `reconfig` command or waiting for the timer configured by the **interface-interval** configuration option of `named.conf(5)`.

secroots [`<view>` ...]

Dump the server's security roots to the `secroots` file for the specified views. If no view is specified, security roots for all views are dumped.

sign `<zone>` [`<class>` [`<view>`]]

Fetch all DNSSEC keys for the given zone from the key directory (see the `key-directory` configuration option of `named.conf(5)`). If they are within their publication period, merge them into the zone's DNSKEY RRset. If the DNSKEY RRset is changed, then the zone is automatically re-signed with the new key set.

This command requires that the **auto-dnssec** zone configuration option of `named.conf(5)` be set to *allow* or *maintain*, and also requires the zone to be configured to allow dynamic updates. (See "Dynamic Update Policies" in the Loop User Manual for more details.)

Also see the `loadkeys` command.

signing (`-list` | `-clear` `<keyid/algorithm>` | `-clear all` | `-nsec3param` (`<parameters>` | `none`)) `<zone>` [

List, edit, or remove the DNSSEC signing state records for the specified zone. The status of ongoing DNSSEC operations (such as signing or generating NSEC3 chains) is stored in the zone in the form of DNS resource records of type *sig-signing-type*.

signing -list converts these records into a human-readable form, indicating which keys are currently signing or have finished signing the zone, and which NSEC3 chains are being created or removed.

signing -clear can remove a single key (specified in the same format that **signing -list** uses to display it), or all keys. In either case, only completed keys are removed; any record indicating that a key has not yet finished signing the zone will be retained.

signing -nsec3param sets the NSEC3 parameters for a zone. This is the only supported mechanism for using NSEC3 with **inline-signing** zones. Parameters are specified in the same format as an NSEC3PARAM resource record: `<hash-algorithm>`, `<flags>`, `<iterations>`, and `<salt>`, in that order.

Currently, the only defined value for `<hash-algorithm>` is `1`, representing SHA-1. `<flags>` may be set to `0` or `1`, depending on whether you wish to set the opt-out bit in the NSEC3 chain. `<iterations>` defines the number of additional times to

apply the algorithm when generating an NSEC3 hash. *<salt>* is a string of data expressed in hexadecimal, a hyphen (-) if no salt is to be used, or the keyword *auto*, which causes *named(8)* to generate a random 64-bit salt.

So, for example, to create an NSEC3 chain using the SHA-1 hash algorithm, no opt-out flag, 10 iterations, and a salt value of *FFFF* for zone *example.org*, use:

```
$ rndc signing -nsec3param 1 0 10 FFFF example.org
```

To set the opt-out flag, 15 iterations, and no salt, use:

```
$ rndc signing -nsec3param 1 1 15 - example.org
```

signing -nsec3param none removes an existing NSEC3 chain and replaces it with NSEC.

stats

Write server statistics to the statistics file (see the *statistics-file* configuration option of *named.conf(5)*).

status

Display status of the server. Note that the number of zones includes the internal *loop/CH* zone and the default *./IN* hint zone if there is not an explicit root zone configured.

stop [-p]

Stop the server, making sure any recent changes made through dynamic update or IXFR are first saved to the master files of the updated zones. If **-p** is specified, the *named(8)* process's PID is returned. This allows an external process to determine when *named(8)* had completed stopping.

Also see the *halt* command.

sync [-clean] [<zone> [<class> [<view>]]]

Sync changes in the journal file for a dynamic zone to the master file. If the *-clean* argument is specified, the journal file is also removed. If no zone is specified, then all zones are synced.

thaw [<zone> [<class> [<view>]]]

Enable updates to a frozen dynamic zone. If no zone is specified, then all frozen zones are enabled. This causes the server to reload the zone from disk, and re-enables dynamic updates after the load has completed. After a zone is thawed, dynamic updates will no longer be refused. If the zone has changed and the **ixfr-from-differences** configuration option of *named.conf(5)* is in use, then the journal file will be updated to reflect changes in the zone. Otherwise, if the zone has changed, any existing journal file will be removed.

Also see the *freeze* command.

trace [<level>]

Sets the nameserver's debugging level to *<level>*. If no *<level>* is specified, the current debugging level is incremented by one.

Also see the *notrace* command.

tsig-delete <keyname> [<view>]

Deletes a given TKEY-negotiated key from the server. (This does not apply to statically configured TSIG keys.)

tsig-list

Lists the names of all TSIG keys currently configured for use by *named(8)* in each view. The list includes both statically configured keys and dynamic TKEY-negotiated keys.

validation (on | off | status) [<view> ...]

Enable, disable, or check the current status of DNSSEC validation. **dnssec-enable** configuration option of *named.conf(5)* also needs to be set to *yes* or *auto* for this command to be effective. It defaults to *on*.

zonestatus <zone> [<class> [<view>]]

Displays the current status of the given <zone>, including the master file name and any include files from which it was loaded, when it was most recently loaded, the current serial number, the number of nodes, whether the zone supports dynamic updates, whether the zone is DNSSEC signed, whether it uses automatic DNSSEC key management or inline signing, and the scheduled refresh or expiry times for the zone.

4.3.5 Files

/etc/loop/rndc.conf

The default configuration file for **rndc**. A complete description of the configuration file is provided in *rndc.conf(5)*.

/etc/loop/rndc.key

The default key used to authenticate commands sent to the server if the configuration file does not exist.

4.3.6 See also

rndc.conf(5), *rndc-confgen(1)*, *named(8)*, *named.conf(5)*

4.3.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2001, 2004-2005, 2007, 2013-2018 Internet Systems Consortium, Inc. ("ISC").

4.4 nsec3hash --- NSEC3 hash generator

4.4.1 Synopsis

nsec3hash <salt> <algorithm> <iterations> <domain>

4.4.2 Description

nsec3hash generates an NSEC3 hash based on a set of NSEC3 parameters. This can be used to check the validity of NSEC3 records in a signed zone.

4.4.3 Options

<salt>

The salt provided to the hash algorithm.

<algorithm>

A number indicating the hash algorithm. Currently the only supported hash algorithm for NSEC3 is SHA-1, which is indicated by the number 1; consequently "1" is the only useful value for this argument.

<iterations>

The number of additional times the hash should be performed.

<domain>

The domain name to be hashed.

4.4.4 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.5 arpaname --- IP address to ARPA name translator

4.5.1 Synopsis

arpaname <ip-address> [<ip-address> ...]

4.5.2 Description

arpaname translates IPv4 and IPv6 address arguments to the corresponding *IN-ADDR.ARPA* or *IP6.ARPA* names.

<ip-address> [<ip-address> ...]

One or more IP addresses to be translated.

4.5.3 See also

dig(1)

4.5.4 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.6 ***rndc-confgen*** --- *rndc(1)* key generator

4.6.1 Synopsis

```
rndc-confgen [-a] [-A <algorithm>] [-b <key-size>] [-c <key-file>] [-h] [-k <key-name>]  
[-p <port>] [-s <address>] [-t <chroot-dir>] [-u <user>]
```

```
rndc-confgen [-h | -V]
```

4.6.2 Description

rndc-confgen generates configuration files for *rndc(1)*. It can be used as a convenient alternative to writing the *rndc.conf* file and the corresponding **controls** and **key** configuration statements of *named(8)* by hand. Alternatively, it can be run with the *-a* option to set up a *rndc.key* file and avoid the need for a *rndc.conf* file and a **controls** statement altogether.

4.6.3 Options

-a

Do automatic ***rndc*** configuration. This creates a file */etc/loop/rndc.key* that is read by both *rndc(1)* and *named(8)* on startup. The */etc/loop/rndc.key* file defines a default command channel and authentication key allowing *rndc(1)* to communicate with *named(8)* on the local host with no further configuration.

If a more elaborate configuration than that generated by the *-a* option is required, for example if *rndc(1)* is to be used remotely, you should run ***rndc-confgen*** without the *-a* option and set up a *rndc.conf* and *named.conf* as directed.

-A <algorithm>

Specifies the algorithm to use for the *rndc* key. Available choices are *hmac-sha256* and *hmac-sha512*. The default is *hmac-sha256*.

-b <key-size>

Specifies the size of the authentication key in bits. It must be between 1 and 512 bits. The default is the hash size.

- c** <key-file>
Used with the **-a** option to specify an alternate location for `rndc.key`.
- h**
Print program usage information and exit.
- k** <key-name>
Specifies the key name of the `rndc` authentication key. This must be a valid domain name. The default is `rndc-key`.
- p** <port>
Specifies the command channel port where `named(8)` listens for connections from `rndc(1)`. The default is 953.
- s** <address>
Specifies the IP address where `named(8)` listens for command channel connections from `rndc(1)`. The default is the loopback address `127.0.0.1`.
- t** <chroot-dir>
Used with the **-a** option to specify a directory where `named(8)` will run `chroot(2)`ed. An additional copy of the `rndc.key` will be written relative to this directory so that it will be found by the `chroot(2)`ed `named(8)` process.
- u** <user>
Used with the **-a** option to set the owner of the `rndc.key` file generated. If **-t** is also specified only the file in the `chroot(2)` area has its owner changed.
- v**
Enable verbose logging.
- V**
Print program version and exit.

4.6.4 Examples

To allow `rndc(1)` to be used with no manual configuration, run:

```
$ rndc-confgen -a
```

To print a sample `rndc.conf` file and corresponding **controls** and **key** statements to be manually inserted into `named.conf(5)`, run:

```
$ rndc-confgen
```

4.6.5 See also

`rndc(1)`, `rndc.conf(5)`, `named(8)`

4.6.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2001, 2003-2005, 2007, 2009, 2013-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.7 `ddns-confgen` --- TSIG key generator

4.7.1 Synopsis

```
ddns-confgen [-a <algorithm>] [-k <key-name>] [-q] [-s <name> | -z <zone>]
```

```
ddns-confgen [-h | -V]
```

4.7.2 Description

ddns-confgen generates keys for use in TSIG signing. The resulting keys can be used, for example, to secure dynamic DNS updates to a zone.

Unless the `-q` option is specified, the generated key is accompanied by configuration text and instructions that can be used with **nsupdate** and **named** when setting up dynamic DNS, including an example **update-policy** statement. This usage is similar to the **rndc-confgen** command for setting up command channel security.

Note that **named** itself can configure a local DDNS key for use with **nsupdate -l**. It does this when a zone is configured with *update-policy local*. **ddns-confgen** is only needed when a more elaborate configuration is required, for instance, if **nsupdate** is to be used from a remote system.

4.7.3 Options

-a <algorithm>

Specifies the algorithm to use for the TSIG key. Available choices are *hmac-md5*, *hmac-sha1*, *hmac-sha224*, *hmac-sha256*, *hmac-sha384* and *hmac-sha512*. The default is *hmac-sha256*. Options are case-insensitive, and the "hmac-" prefix may be omitted.

-h

Print program usage information and exit.

-k <key-name>

Specifies the key name of the DDNS authentication key. The default is *ddns-key* when neither the `-s` nor `-z` options are specified; otherwise, the default is *ddns-key* as a separate label followed by the argument of the option, e.g., *ddns-key.example.com*. The key name must have the format of a valid domain name, consisting of letters, digits, hyphens and periods.

-q

Quiet mode. Print only the key, with no explanatory text or usage examples.

-s <name>

Generate configuration example to allow dynamic updates of a single hostname. The example *named.conf(5)* configuration shows how to set an update policy for the specified name using the "name" nametype. The default key name is *ddns-key.name*. Note that the "self" nametype cannot be used, since the name to be updated may differ from the key name. This option cannot be used with the **-z** option.

-v

Print program version and exit.

-z <zone>

Generate configuration example to allow dynamic updates of a zone: The example *named.conf(5)* configuration shows how to set an update policy for the specified zone using the "zonesub" nametype, allowing updates to all subdomain names within that zone. This option cannot be used with the **-s** option.

4.7.4 See also

nsupdate(1), *named.conf(5)*, *named(8)*

4.7.5 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.8 named-checkconf --- named.conf syntax checker

4.8.1 Synopsis

named-checkconf [-j] [-p [-x]] [-t <directory>] [-z] [<filename>]

named-checkconf [-h | -V]

4.8.2 Description

named-checkconf checks the syntax, but not the semantics, of a *named* configuration file. The file is parsed and checked for syntax errors, along with all files included by it. If no file is specified, */etc/loop/named.conf* is read by default.

Note: files that *named* reads in separate parser contexts, such as *rndc.key*, are not automatically read by *named-checkconf*. Configuration errors in these files may cause *named* to fail to run, even if *named-checkconf* was successful. *named-checkconf* can be run on these files explicitly, however.

4.8.3 Options

-h

Print program usage information and exit.

-j

When loading a zonefile read the journal if it exists.

-p

Print out the `named.conf` and included files in canonical form if no errors were detected. See also the `-x` option.

-t <directory>

chroot (2) to <directory> so that include directives in the configuration file are processed as if run by a similarly *chroot* (2) ed **named**.

-v

Print program version and exit.

-x

When printing the configuration files in canonical form, obscure shared secrets by replacing them with strings of question marks (?). Such configuration files can be shared in bug reports without compromising private data. This option cannot be used without `-p`.

-z

Perform a test load of all master zones found in *named.conf* (5).

<filename>

The name of the configuration file to be checked. If not specified, it defaults to `/etc/loop/named.conf`.

4.8.4 Exit status

named-checkconf returns an exit status of 1 if errors were detected and 0 otherwise.

4.8.5 See also

named.conf (5), *named* (8), *named-checkzone* (1)

4.8.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2002, 2004-2005, 2007, 2009, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.9 `named-checkzone` --- Zone master file syntax checker

4.9.1 Synopsis

named-checkzone [-d] [-j] [-q] [-c class] [-J filename] [-i mode] [-k mode] [-m mode] [-M mode] [-n mode] [-l ttl] [-L serial] [-o filename] [-r mode] [-s style] [-S mode] [-t directory] [-T mode] [-w directory] [-D] [-W mode] <zonename> <filename>

named-checkzone [-h | -V]

4.9.2 Description

named-checkzone checks the syntax and integrity of a zone master file. It performs the same checks as **named** does when loading a zone. This makes **named-checkzone** useful for checking zone files before configuring them to be loaded into a nameserver.

4.9.3 Options

-d

Enable debugging.

-h

Print program usage information and exit.

-q

Quiet mode - exit code only.

-j

When loading a zone file, read the journal if it exists. The journal file name is assumed to be the zone file name appended with the string *.jnl*.

-J <filename>

When loading the zone file read the journal from the given file, if it exists. Using this option implies *-j*.

-c <class>

Specify the class of the zone. If not specified, *IN* is assumed.

-i <mode>

Perform post-load zone integrity checks. Possible <mode> values are *full* (default), *full-sibling*, *local*, *local-sibling* and *none*.

Mode *full* checks that MX records refer to A or AAAA record (both in-zone and out-of-zone hostnames). Mode *local* only checks MX records which refer to in-zone hostnames.

Mode *full* checks that SRV records refer to A or AAAA record (both in-zone and out-of-zone hostnames). Mode *local* only checks SRV records which refer to in-zone hostnames.

Mode *full* checks that delegation NS records refer to A or AAAA record (both in-zone and out-of-zone hostnames). It also checks that glue address records in the zone match those advertised by the child. Mode *local* only checks NS records which refer to in-zone hostnames or that some required glue exists, that is when the nameserver is in a child zone.

Mode *full-sibling* and *local-sibling* disable sibling glue checks but are otherwise the same as *full* and *local* respectively.

Mode *none* disables the checks.

- k** <mode>
Perform **check-names** checks with the specified failure mode. Possible modes are *fail*, *warn*, and *ignore*.
- l** <ttl>
Sets a maximum permissible TTL for the input file. Any record with a TTL higher than this value will cause the zone to be rejected. This is similar to using the **max-zone-ttl** option in *named.conf* (5).
- L** <serial>
When compiling a zone, set the "source serial" value in the header to the specified <serial> number. (This is expected to be used primarily for testing purposes.)
- m** <mode>
Specify whether MX records should be checked to see if they are addresses. Possible modes are *fail*, *warn* (default) and *ignore*.
- M** <mode>
Check if a MX record refers to a CNAME. Possible modes are *fail*, *warn* (default) and *ignore*.
- n** <mode>
Specify whether NS records should be checked to see if they are addresses. Possible modes are *fail*, *warn*, and *ignore*.
- o** <filename>
Write zone output to <filename>. If <filename> is - then write to *stdout* (standard output).
- r** <mode>
Check for records that are treated as different by DNSSEC but are semantically equal in plain DNS. Possible modes are *fail*, *warn* (default) and *ignore*.
- s** <style>
Specify the style of the dumped zone file. Possible <style> values are *full* (default) and *relative*. The full format is most suitable for processing automatically by a separate script. On the other hand, the relative format is more human-readable and is thus suitable for editing by hand. For **named-checkzone** this does not cause any effects unless it dumps the zone contents.

- S** <mode>
Check if a SRV record refers to a CNAME. Possible modes are *fail*, *warn* (default) and *ignore*.
- t** <directory>
chroot (2) to <directory> so that include directives in the configuration file are processed as if run by a similarly *chroot* (2) ed named.
- T** <mode>
Check if SPF records exist and issues a warning if an SPF-formatted TXT record is not also present. Possible modes are *warn* (default), *ignore*.
- V**
Print program version and exit.
- w** <directory>
Change directory to <directory> so that relative filenames in master file **\$INCLUDE** directives work. This is similar to the **directory** clause in *named.conf* (5).
- D**
Dump zone file in canonical format.
- W** <mode>
Specify whether to check for non-terminal wildcards. Non-terminal wildcards are almost always the result of a failure to understand the wildcard matching algorithm ([RFC 1034](#)). Possible modes are *warn* (default) and *ignore*.
- <zonename>**
The domain name of the zone being checked.
- <filename>**
The name of the zone file.

4.9.4 Exit status

named-checkzone returns an exit status of 1 if errors were detected and 0 otherwise.

4.9.5 See also

named (8), *named-checkconf* (1), *named-rrchecker* (1)

4.9.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2002, 2004-2007, 2009-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.10 `named-rrchecker` --- Resource records syntax checker

4.10.1 Synopsis

named-rrchecker [-o <origin>] [-p] [-u] [-C] [-T] [-P]

named-rrchecker [-h | -V]

4.10.2 Description

named-rrchecker reads an individual DNS resource record from standard input (*stdin*) and checks if it is syntactically correct.

All resource record fields except the owner name must be provided as input. The owner name must be omitted.

4.10.3 Options

-h

Print program usage information and exit.

-o <origin>

Specifies the <origin> to be used when interpreting the record.

-p

Prints out the resulting record in canonical form. If there is no canonical form defined then the record will be printed in unknown record format.

-u

Prints out the resulting record in unknown record format ([RFC 3597](#)).

-C

Print the class mnemonic.

-T

Print the (standard) type mnemonic.

-P

Print the private type mnemonic.

-V

Print program version and exit.

4.10.4 Examples

```
$ echo "IN A 127.0.0.1" | named-rrchecker -p
```

4.10.5 See also

named-checkzone (1)

4.10.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2013-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.11 dnssec-dsfromkey --- DS RR generator

4.11.1 Synopsis

dnssec-dsfromkey [-v level] [-a <algorithm>] [-C] [-l domain] [-T <ttd>] <keyfile>

dnssec-dsfromkey [-s] [-a <algorithm>] [-K directory] [-l domain] [-s] [-c class] [-T <ttd>] [-f file] [-A] [-v level] <dnsname>

dnssec-dsfromkey [-h | -V]

4.11.2 Description

dnssec-dsfromkey generates DS and CDS resource records for the given DNSKEYs. By default, records using SHA-256 and SHA-384 digests are printed.

4.11.3 Options

-a <algorithm>

Select the digest algorithm. The value of <algorithm> must be one of *SHA256* (SHA-256), or *SHA384* (SHA-384). These values are case-insensitive.

-C

Generate CDS records rather than DS records. This is mutually exclusive with generating lookaside records.

-T <ttd>

Specifies the TTL of the DS records.

-K <directory>

Look for key files (or, in keyset mode, *keyset*-files) in <directory>.

-f <file>

Zone file mode: in place of the keyfile name, the argument is the DNS domain name of a zone master file, which can be read from <file>. If the zone name is the same as <file>, then it may be omitted.

If <file> is set to -, then the zone data is read from the standard input. This makes it possible to use the output of the *dig(1)* command as input, as in:

```
$ dig dnskey example.com | dnssec-dsfromkey -f - example.
↳ com
```

-A

Include ZSKs when generating DS records. Without this option, only keys which have the KSK flag set will be converted to DS records and printed. Useful only in zone file mode.

-s

Keyset mode: in place of the keyfile name, the argument is the DNS domain name of a keyset file.

-c <class>

Specifies the DNS class. The default is *IN*. Useful only in keyset or zone file mode.

-v <level>

Set the verbosity level.

-h

Print usage information and exit.

-v

Print program version and exit.

4.11.4 Examples

To generate DS RRs for the keyfile `Kexample.com.+013+15010.key`, the following command can be used:

```
$ dnssec-dsfromkey Kexample.com.+013+15010.key
```

The command would print something like:

```
example.com. IN DS 15010 13 2
↳ 9F84AB5A308C7179913242A4FBF158DB6C6258A97D132DB05E1B4E72D1FF9CA0
example.com. IN DS 15010 13 4
↳ A7E3D858C22859DC8708A49EB98C19922C371B4B4DBDCAEEC9E098A85F60B9C78C45108
```

4.11.5 Files

The keyfile can be designed by the key identification *Knnnnn.+aaa+iiii* or the full file name *Knnnnn.+aaa+iiii.key* as generated by *dnssec-keygen (1)*.

The keyset file name is built from the <directory>, the string *keyset-* and the <dnsname>.

4.11.6 Caveat

A keyfile error can result in a "file not found" error even if the file exists.

4.11.7 See also

dnssec-keygen(1), *dnssec-signzone(1)*

4.11.8 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2008-2012, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.12 dnssec-importkey --- DNSKEY importer

4.12.1 Synopsis

dnssec-importkey [-K directory] [-L ttl] [-P date/offset] [-D date/offset] [-v level] <keyfile>

dnssec-importkey [-f filename] [-K directory] [-L ttl] [-P date/offset] [-D date/offset] [-v level] [dnsname]

dnssec-importkey [-h | -V]

4.12.2 Description

dnssec-importkey imports DNSKEY records from external systems so they can be managed. It reads a public DNSKEY record and generates a pair of *.key* and *.private* files. The DNSKEY record may be read from an existing *.key* file, in which case a corresponding *.private* file will be generated, or it may be read from any other file or from the standard input, in which case both *.key* and *.private* files will be generated.

The newly-created *.private* file **does not** contain private key data, and cannot be used for signing. However, having a *.private* file makes it possible to set publication (*-P*) and deletion (*-D*) times for the key, which means the public key can be added to and removed from the DNSKEY RRset on schedule even if the true private key is stored offline.

4.12.3 Options

-f <file>

Zone file mode: instead of a public keyfile name, the argument is the DNS domain name of a zone master file, which can be read from <file>. If the domain name is the same as <file>, then it may be omitted.

If <file> is set to -, then the zone data is read from the standard input.

-K <directory>

Sets the directory in which the key files are to reside.

-L <ttd>

Sets the default TTL to use for this key when it is converted into a DNSKEY RR. If the key is imported into a zone, this is the TTL that will be used for it, unless there was already a DNSKEY RRset in place, in which case the existing TTL would take precedence. Setting the default TTL to 0 or *none* removes it.

-h

Print program usage information and exit.

-v <level>

Set the verbosity level.

-V

Print program version and exit.

4.12.4 Timing options

Dates can be expressed in the format *YYYYMMDD* or *YYYYMMDDHHMMSS*. If the argument begins with a + or -, it is interpreted as an offset from the present time. For convenience, if such an offset is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the offset is computed in years (defined as 365 24-hour days, ignoring leap years), months (defined as 30 24-hour days), weeks, days, hours, or minutes, respectively. Without a suffix, the offset is computed in seconds. To explicitly prevent a date from being set, use *none* or *never*.

-P <date/offset>

Sets the date on which a key is to be published to the zone. After that date, the key will be included in the zone but will not be used to sign it.

-D <date/offset>

Sets the date on which the key is to be deleted. After that date, the key will no longer be included in the zone. (It may remain in the key repository, however.)

4.12.5 Files

A keyfile can be designed by the key identification *Knnnnn.aaa+iiii* or the full file name *Knnnnn.aaa+iiii.key* as generated by *dnssec-keygen (1)*.

4.12.6 See also

dnssec-keygen (1), *dnssec-signzone (1)*

4.12.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2013-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.13 dnssec-keyfromlabel --- DNSKEY importer from HSM

4.13.1 Synopsis

dnssec-keyfromlabel -l <label|uri> [-a <algorithm>] [-A date/offset] [-c class] [-D date/offset] [-f flag] [-G] [-I date/offset] [-i interval] [-K directory] [-L <ttl>] [-n name-type] [-P date/offset] [-p protocol] [-R date/offset] [-S <predecessor-key>] [-T <rrtype>] [-t type] [-v level] [-y] <name>

dnssec-keyfromlabel [-h | -V]

4.13.2 Description

dnssec-keyfromlabel generates a key pair of files that referencing a key object stored in a cryptographic hardware service module (HSM). The private key file can be used for DNSSEC signing of zone data as if it were a conventional signing key created by *dnssec-keygen* (1), but the key material is stored within the HSM, and the actual signing takes place there.

4.13.3 Options

-a <algorithm>

Selects the cryptographic algorithm. For DNSSEC keys, the value of <algorithm> must be one of *RSASHA1*, *NSSEC3RSASHA1*, *RSASHA256*, *RSASHA512*, *ECD-SAP256SHA256*, *ECDSAP384SHA384*, *ED25519* or *ED448*. For KEY keys, the value must be *DH* (Diffie Hellman). These values are case-insensitive.

Specifying this option is mandatory unless **-S** is used. There is no default value. See section 3.4.1 of [RFC 6781](#) for information on selecting an algorithm.

Note

DH automatically sets the **-T KEY** option.

-l <label|uri>

Specifies the label or *pkcs11*: URI for a key pair in the crypto HSM.

The label is an arbitrary string that identifies a particular key. If a label is specified (i.e., without being prefixed with *pkcs11*:), it is internally converted into a URI of the form *pkcs11:object=<label>*.

A *pkcs11*: URI may be directly specified such as:

```
$ dnssec-keyfromlabel -l pkcs11:token=mytoken;object=my-  
→rsa-2048
```

If the URI contains a *pin-source* field, tools using the generated key files will be able to use the HSM for signing and other operations without any need for an

operator to manually enter a PIN. The PIN may also be specified in the OpenSSL config file `openssl.cnf` as part of the `pkcs11-provider` configuration.

Warning

Making the HSM's PIN accessible in this manner may reduce the security advantage of using an HSM. Be sure this is what you want to do before making use of this feature.

-n `<name-type>`

Specifies the owner type of the key. The value of `<name-type>` must either be *ZONE* (for a DNSSEC zone key (KEY/DNSKEY)), *HOST* or *ENTITY* (for a key associated with a host (KEY)), *USER* (for a key associated with a user(KEY)) or *OTHER* (DNSKEY). These values are case-insensitive.

-c `<class>`

Indicates that the DNS record containing the key should have the specified class. If not specified, class *IN* is used.

-f `<flag>`

Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flags are *KSK* for Secure Entry Point (SEP) defined in [RFC 3757](#) and [RFC 4034](#), and *REVOKE* defined in [RFC 5011](#).

-G

Generate a key, but do not publish it or sign with it. This option is incompatible with `-P` and `-A`.

-h

Print program usage information and exit.

-K `<directory>`

Sets the directory in which the key files are to be written.

-L `<ttd>`

Sets the default TTL to use for this key when it is converted into a DNSKEY RR. If the key is imported into a zone, this is the TTL that will be used for it, unless there was already a DNSKEY RRset in place, in which case the existing TTL would take precedence. If this value is not set and there is no existing DNSKEY RRset, the TTL will default to the SOA TTL. Setting the default TTL to *0* or *none* is the same as leaving it unset.

-p `<protocol>`

Sets the protocol value for the key. The protocol is a number between *0* and *255*. The default is *3* (DNSSEC). Other possible values for this argument are listed in [RFC 2535](#) and its successors.

-S `<predecessor-key>`

Create a new key as an explicit successor to an existing `<predecessor-key>`. The name, algorithm, size, and type of the new key will be set to match the existing

key. The activation date of the new key will be set to the inactivation date of the existing one. The publication date will be set to the activation date minus the pre-publication interval, which defaults to 30 days.

-T *<rrtype>*

Specifies the resource record type to use for the key. *<rrtype>* must be either *DNSKEY* or *KEY*. The default is *DNSKEY* when using a DNSSEC algorithm, but it can be overridden to *KEY* for use with SIG(0).

Using the *DH* algorithm forces this option to *KEY*.

-t *<type>*

Indicates the use of the key. *<type>* must be one of *AUTHCONF*, *NOAUTHCONF*, *NOAUTH*, or *NOCONF*. The default is *AUTHCONF*. *AUTH* refers to the ability to authenticate data, and *CONF* the ability to encrypt data.

-v *<level>*

Set the verbosity level.

-V

Print program version and exit.

-y

Allows DNSSEC key files to be generated even if the key ID would collide with that of an existing key, in the event of either key being revoked.

Warning

This is not safe to use if you will be using **RFC 5011** trust anchor maintenance with either of the keys involved.

<name>

The name of the key. This must match the name of the zone for which the key is being generated.

4.13.4 Timing options

Dates can be expressed in the format *YYYYMMDD* or *YYYYMMDDHHMMSS*. If the argument begins with a + or -, it is interpreted as an offset from the present time. For convenience, if such an offset is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the offset is computed in years (defined as 365 24-hour days, ignoring leap years), months (defined as 30 24-hour days), weeks, days, hours, or minutes, respectively. Without a suffix, the offset is computed in seconds. To explicitly prevent a date from being set, use *none* or *never*.

-P *<date/offset>*

Sets the date on which a key is to be published to the zone. After that date, the key will be included in the zone but will not be used to sign it. If not set, and if the *-G* option has not been used, the default is *now*.

-D <date/offset>

Sets the date on which the key is to be deleted. After that date, the key will no longer be included in the zone. (It may remain in the key repository, however.)

-A <date/offset>

Sets the date on which the key is to be activated. After that date, the key will be included in the zone and used to sign it. If not set, and if the **-G** option has not been used, the default is *now*. If set, and if **-P** is not set, then the publication date will be set to the activation date minus the prepublication interval.

-R <date/offset>

Sets the date on which the key is to be revoked. After that date, the key will be flagged as revoked. It will be included in the zone and will be used to sign it.

-I <date/offset>

Sets the date on which the key is to be retired. After that date, the key will still be included in the zone, but it will not be used to sign it.

-i <interval>

Sets the pre-publication interval for a key. If set, then the publication and activation dates must be separated by at least this much time. If the activation date is specified but the publication date isn't, then the publication date will default to this much time before the activation date; conversely, if the publication date is specified but activation date isn't, then activation will be set to this much time after publication.

If the key is being created as an explicit successor to another key, then the default prepublication interval is 30 days; otherwise it is zero.

As with date offsets, if the argument is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the interval is measured in years, months, weeks, days, hours, or minutes, respectively. Without a suffix, the interval is measured in seconds.

4.13.5 Generated key files

When **dnssec-keyfromlabel** completes successfully, it prints a string of the form *Knnnn.+aaa+iiii* to the standard output. This is an identification string for the key files it has generated.

- <nnnn> is the key name.
- <aaa> is the numeric representation of the algorithm.
- <iiii> is the key identifier (or footprint).

dnssec-keyfromlabel creates two files, with names based on the printed string. *Knnnn.+aaa+iiii.key* contains the public key, and *Knnnn.+aaa+iiii.private* contains the private key.

The *.key* file contains a DNSKEY record that can be inserted into a zone file (directly or with a \$INCLUDE statement).

The *.private* file contains algorithm-specific fields. For obvious security reasons, this file does not have general read permission.

4.13.6 See also

dnssec-keygen(1), *dnssec-signzone(1)*

4.13.7 Copyright

Copyright (C) 2024, 2025 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2008-2012, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.14 dnssec-keygen --- DNSKEY and KEY generator

4.14.1 Synopsis

dnssec-keygen [-a <algorithm>] [-b <key-size>] [-n name-type] [-A date/offset] [-c class] [-D date/offset] [-E <password>] [-e] [-f flag] [-G] [-g generator] [-I date/offset] [-i interval] [-K directory] [-l <label | uri>] [-L <ttl>] [-P date/offset] [-p protocol] [-q] [-R date/offset] [-S <predecessor-key>] [-s strength] [-T <rrtype>] [-t type] [-v level] [-z <name>]

dnssec-keygen [-h | -V]

4.14.2 Description

dnssec-keygen generates keys for DNSSEC (Secure DNS), as defined in [RFC 2535](#) and [RFC 4034](#). It can also generate keys for use with TSIG (Transaction Signatures) as defined in [RFC 2845](#), or TKEY (Transaction Key) as defined in [RFC 2930](#).

4.14.3 Options

-a <algorithm>

Selects the cryptographic algorithm. For DNSSEC keys, the value of <algorithm> must be one of *RSASHA1*, *NSEC3RSASHA1*, *RSASHA256*, *RSASHA512*, *ECDSAP256SHA256*, *ECDSAP384SHA384*, *ED25519* or *ED448*. For TSIG/TKEY, the value must be *DH* (Diffie Hellman), *HMAC-MD5*, *HMAC-SHA1*, *HMAC-SHA224*, *HMAC-SHA256*, *HMAC-SHA384*, or *HMAC-SHA512*. These values are case-insensitive.

Specifying this option is mandatory unless **-S** is used. There is no default value. See section 3.4.1 of [RFC 6781](#) for information on selecting an algorithm.

Note

DH, *HMAC-MD5*, *HMAC-SHA1*, *HMAC-SHA224*, *HMAC-SHA256*, *HMAC-SHA384*, and *HMAC-SHA512* automatically set the **-T KEY** option.

-b <key-size>

Specifies the number of bits in the key. The choice of key size depends on the algorithm used. RSA keys must be between 512 and 4096 bits. Diffie Hellman keys must be between 128 and 4096 bits. HMAC keys must be between 1 and 512 bits. Elliptic curve algorithms don't need this parameter.

Specifying this option is mandatory for algorithms except ECDSAP256SHA256, ECDSAP384SHA384, ED25519, and ED448 where there is no default value. See section 3.4.2 of [RFC 6781](#) for information on selecting a key size.

-n <name-type>

Specifies the owner type of the key. The value of <name-type> must either be *ZONE* (for a DNSSEC zone key (KEY/DNSKEY)), *HOST* or *ENTITY* (for a key associated with a host (KEY)), *USER* (for a key associated with a user(KEY)) or *OTHER* (DNSKEY). These values are case-insensitive. Defaults to *ZONE* for DNSKEY generation.

-c <class>

Indicates that the DNS record containing the key should have the specified class. If not specified, class *IN* is used.

-e

Encrypt the private key. A password may be specified using the *-E* option, otherwise, it is prompted for. The encrypted private key is written to a separate *.pem* file (see the section titled *Generated key files*).

-E <password>

The password to encrypt the private key with. If this option is not specified, a password is prompted for if **dnssec-keygen** is run interactively. The encrypted private key is written to a separate *.pem* file (see the section titled *Generated key files*).

Warning

Be careful in providing passwords using this command line argument, and saving them in scripts. You would have to protect these scripts somehow. Providing passwords interactively when prompted for is safer, but that may not work for non-interactive uses.

-f <flag>

Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flags are *KSK* for Secure Entry Point (SEP) defined in [RFC 3757](#) and [RFC 4034](#), and *REVOKE* defined in [RFC 5011](#).

-G

Generate a key, but do not publish it or sign with it. This option is incompatible with *-P* and *-A*.

-g <generator>

If generating a Diffie Hellman key, use this generator. Allowed `<generator>` values are 2 and 5. If no generator is specified, a known prime from [RFC 2539](#) will be used if possible; otherwise the default is 2.

-K `<directory>`

Sets the directory in which the key files are to be written.

-l `<label|uri>`

Specifies the label or `pkcs11:` URI to generate a key pair within the crypto HSM.

The label is an arbitrary string that identifies a particular key. If a label is specified (i.e., without being prefixed with `pkcs11:`), it is internally converted into a URI of the form `pkcs11:object=<label>`.

A `pkcs11:` URI may be directly specified such as:

```
$ dnssec-keygen -l pkcs11:token=mytoken;object=my-rsa-2048
```

If the URI contains a *pin-source* field, tools using the generated key files will be able to use the HSM for signing and other operations without any need for an operator to manually enter a PIN. The PIN may also be specified in the OpenSSL config file `openssl.cnf` as part of the `pkcs11-provider` configuration.

Warning

Making the HSM's PIN accessible in this manner may reduce the security advantage of using an HSM. Be sure this is what you want to do before making use of this feature.

-L `<ttd>`

Sets the default TTL to use for this key when it is converted into a DNSKEY RR. If the key is imported into a zone, this is the TTL that will be used for it, unless there was already a DNSKEY RRset in place, in which case the existing TTL would take precedence. If this value is not set and there is no existing DNSKEY RRset, the TTL will default to the SOA TTL. Setting the default TTL to 0 or *none* is the same as leaving it unset.

-P `<protocol>`

Sets the protocol value for the key. The protocol is a number between 0 and 255. The default is 3 (DNSSEC). Other possible values for this argument are listed in [RFC 2535](#) and its successors.

-q

Quiet mode: Suppresses unnecessary output, including progress indication. Without this option, when **dnssec-keygen** is run interactively to generate an RSA or DSA key pair, it will print a string of symbols to standard-error indicating the progress of the key generation. A dot(.) indicates that a random number has been found which passed an initial sieve test; plus(+) means a number has

passed a single round of the Miller-Rabin primality test; a space means that the number has passed all the tests and is a satisfactory key.

-S *<predecessor-key>*

Create a new key as an explicit successor to an existing *<predecessor-key>*. The name, algorithm, size, and type of the new key will be set to match the existing key. The activation date of the new key will be set to the inactivation date of the existing one. The publication date will be set to the activation date minus the pre-publication interval, which defaults to 30 days.

-s *<strength>*

Specifies the strength value of the key. The strength is a number between 0 and 15, and currently has no defined purpose in DNSSEC.

-T *<rrtype>*

Specifies the resource record type to use for the key. *<rrtype>* must be either *DNSKEY* or *KEY*. The default is *DNSKEY* when using a DNSSEC algorithm, but it can be overridden to *KEY* for use with SIG(0).

Using any TSIG algorithm (*HMAC-** or *DH*) forces this option to *KEY*.

-t *<type>*

Indicates the use of the key. *<type>* must be one of *AUTHCONF*, *NOAUTHCONF*, *NOAUTH*, or *NOCONF*. The default is *AUTHCONF*. *AUTH* refers to the ability to authenticate data, and *CONF* the ability to encrypt data.

-h

Print program usage information and exit.

-v *<level>*

Set the verbosity level.

-V

Print program version and exit.

<name>

The name of the key. For DNSSEC keys, this must match the name of the zone for which the key is being generated.

4.14.4 Timing options

Dates can be expressed in the format *YYYYMMDD* or *YYYYMMDDHHMMSS*. If the argument begins with a + or -, it is interpreted as an offset from the present time. For convenience, if such an offset is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the offset is computed in years (defined as 365 24-hour days, ignoring leap years), months (defined as 30 24-hour days), weeks, days, hours, or minutes, respectively. Without a suffix, the offset is computed in seconds. To explicitly prevent a date from being set, use *none* or *never*.

-P *<date/offset>*

Sets the date on which a key is to be published to the zone. After that date, the

key will be included in the zone but will not be used to sign it. If not set, and if the `-G` option has not been used, the default is *now*.

-D `<date/offset>`

Sets the date on which the key is to be deleted. After that date, the key will no longer be included in the zone. (It may remain in the key repository, however.)

-A `<date/offset>`

Sets the date on which the key is to be activated. After that date, the key will be included in the zone and used to sign it. If not set, and if the `-G` option has not been used, the default is *now*. If set, and if `-P` is not set, then the publication date will be set to the activation date minus the prepublication interval.

-R `<date/offset>`

Sets the date on which the key is to be revoked. After that date, the key will be flagged as revoked. It will be included in the zone and will be used to sign it.

-I `<date/offset>`

Sets the date on which the key is to be retired. After that date, the key will still be included in the zone, but it will not be used to sign it.

-i `<interval>`

Sets the pre-publication interval for a key. If set, then the publication and activation dates must be separated by at least this much time. If the activation date is specified but the publication date isn't, then the publication date will default to this much time before the activation date; conversely, if the publication date is specified but activation date isn't, then activation will be set to this much time after publication.

If the key is being created as an explicit successor to another key, then the default prepublication interval is 30 days; otherwise it is zero.

As with date offsets, if the argument is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the interval is measured in years, months, weeks, days, hours, or minutes, respectively. Without a suffix, the interval is measured in seconds.

4.14.5 Generated key files

When **dnssec-keygen** completes successfully, it prints a string of the form *Knnnn.+aaa+iiii* to the standard output. This is an identification string for the key files it has generated.

- *<nnnn>* is the key name.
- *<aaa>* is the numeric representation of the algorithm.
- *<iiii>* is the key identifier (or footprint).

dnssec-keygen creates two or three files, with names based on the printed string. *Knnnn.+aaa+iiii.key* contains the public key, *Knnnn.+aaa+iiii.private* contains the private key and metadata, and *Knnnn.+aaa+iiii.pem* contains the encrypted private key.

The *.key* file contains a DNSKEY record that can be inserted into a zone file (directly or with a \$INCLUDE statement).

The *.private* file contains algorithm-specific fields. For obvious security reasons, this file does not have general read permission.

The *.pem* file contains the encrypted private key if encryption was requested. If the private key is to be encrypted, the encrypted private key is written to a file named *Knnnnn.+aaa+iiii.pem*, and the *Knnnnn.+aaa+iiii.private* file contains just metadata about the private key.

Both *.key* and *.private* files are generated even for symmetric cryptography algorithms such as *HMAC-MD5*, even though the public and private key are equivalent.

4.14.6 Examples

To generate a DNSKEY for the domain *example.com*, the following command would be issued:

```
[user@host ~]$ dnssec-keygen -a ECDSAP256SHA256 example.com
Generating key pair.
Kexample.com.+013+32634
[user@host ~]$
```

The command would print a string of the form:

```
Kexample.com.+013+32634
```

In this example, **dnssec-keygen** creates the files *Kexample.com.+013+32634.key* and *Kexample.com.+013+32634.private*, containing a key of the specified *ECDSAP256SHA256* algorithm type.

4.14.7 See also

dnssec-signzone (1), *dnssec-keyfromlabel (1)*

4.14.8 Copyright

Copyright (C) 2024, 2025 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2005, 2007-2012, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.15 dnssec-revoke --- DNSKEY revoker

4.15.1 Synopsis

dnssec-revoke [-v level] [-r] [-K directory] [-E <password>] [-f] [-R] <keyfile>

dnssec-revoke [-h | -V]

4.15.2 Description

dnssec-revoke reads a DNSSEC key file, sets the *REVOKE* bit on the key as defined in [RFC 5011](#), and creates a new pair of key files containing the now-revoked key.

4.15.3 Options

- K** <directory>
Sets the directory in which the key files are to reside.
- r**
After writing the new keyset files remove the original keyset files.
- E** <password>
Specify the password for the private key. If the password is incorrect and does not decrypt a private key, the password is prompted for.
- f**
Force overwrite. Causes **dnssec-revoke** to write the new key pair even if a file already exists matching the algorithm and key ID of the revoked key.
- R**
Print the key tag of the key with the *REVOKE* bit set but do not revoke the key.
- h**
Print program usage information and exit.
- v** <level>
Set the verbosity level.
- V**
Print program version and exit.

4.15.4 See also

dnssec-keygen(1)

4.15.5 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009, 2011, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.16 dnssec-settime --- DNSKEY timing metadata setter

4.16.1 Synopsis

dnssec-settime [-E <password>] [-f] [-K directory] [-L ttl] [-P date/offset] [-A date/offset] [-R date/offset] [-I date/offset] [-D date/offset] [-S <predecessor-key>] [-i interval] [-v level] <keyfile>

dnssec-settime [-h | -V]

4.16.2 Description

dnssec-settime reads a DNSSEC private key file and sets the key timing metadata as specified by the *-P*, *-A*, *-R*, *-I*, and *-D* options. The metadata can then be used by *dnssec-signzone*(1) or other signing software to determine when a key is to be published, whether it should be used for signing a zone, etc.

If none of these options is set on the command line, then **dnssec-settime** simply prints the key timing metadata already stored in the key.

When key metadata fields are changed, both files of a key pair (*Knnnn.aaa+iiii.key* and *Knnnn.aaa+iiii.private*) are regenerated. Metadata fields are stored in the private file. A human-readable description of the metadata is also placed in comments in the key file. The private file's permissions are always set to be inaccessible to anyone other than the owner (mode 0600).

4.16.3 Options

-E <password>

Specify the password for the private key. If the password is incorrect and does not decrypt a private key, the password is prompted for.

-f

Force an update of an old-format key with no metadata fields. Without this option, **dnssec-settime** will fail when attempting to update a legacy key. With this option, the key will be recreated in the new format, but with the original key data retained. The key's creation date will be set to the present time. If no other values are specified, then the key's publication and activation dates will also be set to the present time.

Error

TODO: Remove this option.

-K <directory>

Sets the directory in which the key files are to reside.

-L <t1>

Sets the default TTL to use for this key when it is converted into a DNSKEY RR. If the key is imported into a zone, this is the TTL that will be used for it, unless there was already a DNSKEY RRset in place, in which case the existing TTL would take precedence. If this value is not set and there is no existing DNSKEY RRset, the TTL will default to the SOA TTL. Setting the default TTL to 0 or *none* removes it from the key.

-h

Print program usage information and exit.

-v <level>

Set the verbosity level.

-V

Print program version and exit.

4.16.4 Timing options

Dates can be expressed in the format *YYYYMMDD* or *YYYYMMDDHHMMSS*. If the argument begins with a *+* or *-*, it is interpreted as an offset from the present time. For convenience, if such an offset is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the offset is computed in years (defined as 365 24-hour days, ignoring leap years), months (defined as 30 24-hour days), weeks, days, hours, or minutes, respectively. Without a suffix, the offset is computed in seconds. To explicitly prevent a date from being set, use *none* or *never*.

-P <date/offset>

Sets the date on which a key is to be published to the zone. After that date, the key will be included in the zone but will not be used to sign it.

-D <date/offset>

Sets the date on which the key is to be deleted. After that date, the key will no longer be included in the zone. (It may remain in the key repository, however.)

-A <date/offset>

Sets the date on which the key is to be activated. After that date, the key will be included in the zone and used to sign it. If set, and if **-P** is not set, then the publication date will be set to the activation date minus the prepublication interval.

-R <date/offset>

Sets the date on which the key is to be revoked. After that date, the key will be flagged as revoked. It will be included in the zone and will be used to sign it.

-I <date/offset>

Sets the date on which the key is to be retired. After that date, the key will still be included in the zone, but it will not be used to sign it.

-i <interval>

Sets the pre-publication interval for a key. If set, then the publication and activation dates must be separated by at least this much time. If the activation date is specified but the publication date isn't, then the publication date will default to this much time before the activation date; conversely, if the publication date is specified but activation date isn't, then activation will be set to this much time after publication.

If the key is being created as an explicit successor to another key, then the default prepublication interval is 30 days; otherwise it is zero.

As with date offsets, if the argument is followed by one of the suffixes *y*, *mo*, *w*, *d*, *h*, or *mi*, then the interval is measured in years, months, weeks, days, hours, or minutes, respectively. Without a suffix, the interval is measured in seconds.

-S <predecessor-key>

Select a key for which the key being modified will be an explicit successor. The name, algorithm, size, and type of the predecessor key must exactly match those of the key being modified. The activation date of the successor key will be set to the inactivation date of the predecessor. The publication date will be set to the activation date minus the prepublication interval, which defaults to 30 days.

4.16.5 Printing options

dnssec-settime can also be used to print the timing metadata associated with a key.

-u

Print times in UNIX epoch format.

-p (C | P | A | R | I | D | all)

Print a specific metadata value or set of metadata values. The **-p** option may be followed by one or more of the following letters to indicate which value or values to print: *C* for the creation date, *P* for the publication date, *A* for the activation date, *R* for the revocation date, *I* for the inactivation date, or *D* for the deletion date. To print all of the metadata, use **-pall**.

4.16.6 See also

dnssec-keygen(1), *dnssec-signzone(1)*

4.16.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2009-2011, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.17 dnssec-signzone --- DNSSEC zone signer

4.17.1 Synopsis

dnssec-signzone [-a] [-c class] [-d directory] [-D] [-e end-time] [-E <password>] [-f output-file] [-g] [-i interval] [-j jitter] [-K directory] [-k key] [-L serial] [-M max-ttl] [-N soa-serial-format] [-n <num-threads>] [-o origin] [-O output-style] [-P] [-R] [-S] [-s start-time] [-T ttl] [-t] [-u] [-v level] [-X <extended-end-time>] [-x] [-z] [-3 salt] [-H iterations] [-A] <zonefile> [key...]

dnssec-signzone [-h | -V]

4.17.2 Description

dnssec-signzone signs a zone. It generates NSEC and RRSIG records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a *keyset* file for each child zone.

4.17.3 Options

- a**
Verify all generated signatures.
- c** <class>
Specifies the DNS class of the zone.
- C**
Compatibility mode: Generate a `keyset-zonename` file in addition to `dsset-zonename` when signing a zone, for use by older versions of **dnssec-signzone**.
- d** <directory>
Look for *dsset-* or *keyset-* files in <directory>.
- D**
Output only those record types automatically managed by **dnssec-signzone**, i.e. RRSIG, NSEC, NSEC3 and NSEC3PARAM records. If smart signing (*-S*) is used, DNSKEY records are also included. The resulting file can be included in the original zone file with the \$INCLUDE master file directive. This option cannot be combined with serial number updating.
- g**
Generate DS records for child zones from *dsset-* or *keyset-* file. Existing DS records will be removed.
- K** <directory>
Key repository: Specify a directory to search for DNSSEC keys. If not specified, defaults to the current directory.
- E** <password>
Specify the password for the private key. If the password is incorrect and does not decrypt a private key, the password is prompted for.
- k** <key>
Treat specified <key> as a key signing key (KSK) ignoring any key flags. This option may be specified multiple times.
- M** <max-ttl>
Sets the maximum TTL for the signed zone. Any TTL higher than maxttl in the input zone will be reduced to maxttl in the output. This provides certainty as to the largest possible TTL in the signed zone, which is useful to know when rolling keys because it is the longest possible time before signatures that have been retrieved by resolvers will expire from resolver caches. Zones that are signed with this option should be configured to use a matching **max-zone-ttl** config option in *named.conf(5)*.

Note

This option is incompatible with `-D`, because it modifies non-DNSSEC data in the output zone.

-s <start-time>

Specify the date and time when the generated RRSIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by +N, which is N seconds from the current time. If `-s` option is not specified, the current time minus 1 hour (to allow for clock skew) is used.

-e <end-time>

Specify the date and time when the generated RRSIG records expire. As with `-s`, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with +N, which is N seconds from the start time. A time relative to the current time is indicated with now+N. If `-e` option is not specified, 30 days from the start time is used as a default. <end-time> must be later than <start-time> (either specified with `-s` or its default value).

-X <extended-end-time>

Specify the date and time when the generated RRSIG records for the DNSKEY RRset will expire. This is to be used in cases when the DNSKEY signatures need to persist longer than signatures on other records; e.g., when the private component of the KSK is kept offline and the KSK signature is to be refreshed manually.

As with `-s`, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with +N, which is N seconds from the start time. A time relative to the current time is indicated with now+N. If `-X` is not specified, the value of <end-time> (either specified with `-e` or its default value) is used as the default. <extended-end-time> must be later than <start-time> (either specified with `-s` or its default value).

-f <output-file>

The name of the output file containing the signed zone. The default is to append *.signed* to the input filename. If <output-file> is set to -, then the signed zone is written to standard-output, with a default output format of *full*.

-i <interval>

When a previously-signed zone is passed as input, records may be resigned. The <interval> value specifies the cycle interval as an offset from the current time (in seconds). If an RRSIG record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon, and it will be replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. So if neither `-e` or `-s` are specified, **dnssec-signzone** generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Therefore, if any existing RRSIG records are due to expire in less than 7.5 days, they would be replaced.

-j <jitter>

When signing a zone with a fixed signature lifetime, all RRSIG records issued at the time of signing expire simultaneously. If the zone is incrementally signed, i.e. a previously-signed zone is passed as input to the signer, all expired signatures have to be regenerated at about the same time. The <jitter> value specifies a jitter window that will be used to randomize the signature expire time, thus spreading incremental signature regeneration over time.

Signature lifetime jitter, also to some extent, benefits validators and servers by spreading out cache expiration, i.e. if large numbers of RRSIGs don't expire at the same time from all caches there will be less congestion than if all validators need to refetch at mostly the same time.

-L <serial>

When writing a signed zone, set the "source serial" value in the header to the specified <serial> number. (This is expected to be used primarily for testing purposes.)

-n <num-threads>

Specifies the number of threads to use. By default, one thread is started for each detected processor.

-N <soa-serial-format>

The SOA serial number format of the signed zone. Possible formats are *keep*, *increment*, and *unixtime*.

- *keep* --- Do not modify the SOA serial number
- *increment* --- Increment the SOA serial number using [RFC 1982](#) arithmetic
- *unixtime* --- Set the SOA serial number to the number of seconds since epoch

The default is *keep*.

-o <origin>

The zone origin. If not specified, the name of the zone file is assumed to be the origin.

-O <output-style>

The style of the output file containing the signed zone. Possible formats are *text* and *full*.

- *text* --- the standard textual representation of the zone
- *full* --- text output in a format suitable for processing by external scripts

The default is *text*.

-P

Disable post sign verification tests.

The post sign verification test ensures that for each algorithm in use there is at least one non revoked self signed KSK key, that all revoked KSK keys are self

signed, and that all records in the zone are signed by the algorithm. This option skips these tests.

-Q

Remove signatures from keys that are no longer active.

Normally, when a previously-signed zone is passed as input to the signer, and a DNSKEY record has been removed and replaced with a new one, signatures from the old key that are still within their validity period are retained. This allows the zone to continue to validate with cached copies of the old DNSKEY RRset. The **-Q** option forces **dnssec-signzone** to remove signatures from keys that are no longer active. This enables ZSK rollover using the procedure described in [RFC 6781](#), section 4.1.1.1 (*Pre-Publish Zone Signing Key Rollover*).

-R

Remove signatures from keys that are no longer published.

This option is similar to **-Q**, except it forces **dnssec-signzone** to signatures from keys that are no longer published. This enables ZSK rollover using the procedure described in [RFC 6781](#), section 4.1.1.2 (*Double-Signature Zone Signing Key Rollover*).

-S

Smart signing. Instructs **dnssec-signzone** to search the key repository for keys that match the zone being signed, and to include them in the zone if appropriate.

When a key is found, its timing metadata is examined to determine how it should be used, according to the following rules. Each successive rule takes priority over the prior ones:

1. If no timing metadata has been set for the key, the key is published in the zone and used to sign the zone.
2. If the key's publication date is set and is in the past, the key is published in the zone.
3. If the key's activation date is set and in the past, the key is published (regardless of publication date) and used to sign the zone.
4. If the key's revocation date is set and in the past, and the key is published, then the key is revoked, and the revoked key is used to sign the zone.
5. If either of the key's unpublication or deletion dates are set and in the past, the key is NOT published or used to sign the zone, regardless of any other metadata.

-T <tttl>

Specifies a TTL to be used for new DNSKEY records imported into the zone from the key repository. If not specified, the default is the TTL value from the zone's SOA record. This option is ignored when signing without **-S**, since DNSKEY records are not imported from the key repository in that case. It is also ignored if there are any pre-existing DNSKEY records at the zone apex, in which case new

records' TTL values will be set to match them, or if any of the imported DNSKEY records had a default TTL value. In the event of a conflict between TTL values in imported keys, the shortest one is used.

-t

Print statistics at completion.

-u

Update NSEC/NSEC3 chain when re-signing a previously signed zone. With this option, a zone signed with NSEC can be switched to NSEC3, or a zone signed with NSEC3 can be switch to NSEC or to NSEC3 with different parameters. Without this option, **dnssec-signzone** will retain the existing chain when re-signing.

-x

Only sign the DNSKEY RRset with key-signing keys (KSKs), and omit signatures from zone-signing keys. This is similar to the **dnssec-dnskey-kskonly** zone config option in *named.conf(5)*.

-z

Ignore KSK flag on key when determining what to sign. This causes KSK-flagged keys to sign all records, not just the DNSKEY RRset. This is similar to the **update-check-ksk** zone config option in *named.conf(5)*.

-3 <salt>

Generate an NSEC3 chain with the given hex encoded salt. A dash (-) value for <salt> can be used to indicate that no salt is to be used when generating the NSEC3 chain.

-H <iterations>

When generating an NSEC3 chain, use this many iterations. The default is 10.

-A

When generating an NSEC3 chain set the OPTOUT flag on all NSEC3 records and do not generate NSEC3 records for insecure delegations.

Using this option twice (i.e., **-AA**) turns the OPTOUT flag off for all records. This is useful when using the **-u** option to modify an NSEC3 chain which previously had OPTOUT set.

Error

TODO: Use a separate option for **-AA** (**-A** specified twice) and remove this hack.

-h

Print program usage information and exit.

-v <level>

Set the verbosity level.

-v

Print program version and exit.

<zonefile>

The file containing the zone to be signed.

[<key> ...]

Specify which keys should be used to sign the zone. If no keys are specified, then the zone will be examined for DNSKEY records at the zone apex. If these are found and there are matching private keys in the current directory, then these will be used for signing.

4.17.4 Examples

The following command signs the *example.com* zone with the *ECDSAP256SHA256* key generated by *dnssec-keygen* (1) (*Kexample.com.+013+17247*). Because the *-s* option is not being used, the zone's keys must be in the master file (*db.example.com*). This invocation looks for *dsset* files, in the current directory, so that DS records can be imported from them (*-g*).

```
% dnssec-signzone -g -o example.com db.example.com Kexample.  
com.+013+17247  
db.example.com.signed  
%
```

In the above example, **dnssec-signzone** creates the file *db.example.com.signed*. This file should be referenced in a zone statement in a *named.conf* (5) configuration file.

The following example re-signs a previously signed zone with default parameters. The private keys are assumed to be in the current directory.

```
% cp db.example.com.signed db.example.com  
% dnssec-signzone -o example.com db.example.com  
db.example.com.signed  
%
```

4.17.5 See also

dnssec-keygen (1)

4.17.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2009, 2011-2018 Internet Systems Consortium, Inc. ("ISC").

4.18 `dnssec-verify` --- DNSSEC zone signature verifier

4.18.1 Synopsis

```
dnssec-verify [-c <class>] [-o <origin>] [-x] [-z] [-v <level>] <zonefile>  
dnssec-verify [-h | -V]
```

4.18.2 Description

dnssec-verify verifies that a zone is fully signed for each algorithm found in the DNSKEY RRset for the zone, and that the NSEC/NSEC3 chains are complete.

4.18.3 Options

-c <class>

Specifies the DNS class of the zone.

-o <origin>

The zone origin. If not specified, the name of the zone file is assumed to be the origin.

-x

Only verify that the DNSKEY RRset is signed with key-signing keys. Without this flag, it is assumed that the DNSKEY RRset will be signed by all active keys. When this flag is set, it will not be an error if the DNSKEY RRset is not signed by zone-signing keys. This corresponds to the `-x` option of *dnssec-signzone(1)*.

-z

Ignore the KSK flag on the keys when determining whether the zone is correctly signed. Without this flag it is assumed that there will be a non-revoked, self-signed DNSKEY with the KSK flag set for each algorithm and that RRsets other than DNSKEY RRset will be signed with a different DNSKEY without the KSK flag set.

With this flag set, we only require that for each algorithm, there will be at least one non-revoked, self-signed DNSKEY, regardless of the KSK flag state, and that other RRsets will be signed by a non-revoked key for the same algorithm that includes the self-signed key; the same key may be used for both purposes. This corresponds to the `-z` option of *dnssec-signzone(1)*.

-h

Print program usage information and exit.

-v <level>

Set the verbosity level.

-v

Print program version and exit.

<zonefile>

The file containing the zone to be signed.

4.18.4 See also

dnssec-signzone (1)

4.18.5 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2012, 2014-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

4.19 **delv** --- DNSSEC lookup and validation utility

4.19.1 Synopsis

```
delv [@<server>] [[-4] | [-6]] [-a <anchor-file>] [-b <address>[#<port>]] [-c <class>] [-d
<level>] [-i] [-m] [-p <port>] [-q <name>] [-t <type>] [-x <address>] [<name>] [<type>]
[<class>] [<query-option> ...]
```

```
delv [-h | -V]
```

4.19.2 Description

delv is a tool for sending DNS queries and validating the results, using the same internal resolver and validator logic as *named (8)*.

delv will send to a specified name server all queries needed to fetch and validate the requested data; this includes the original requested query, subsequent queries to follow CNAME or DNAME chains, and queries for DNSKEY and DS records to establish a chain of trust for DNSSEC validation. It does not perform iterative resolution, but simulates the behavior of a name server configured for DNSSEC validating and forwarding.

By default, responses are validated using built-in DNSSEC trust anchor for the root zone (.). Records returned by **delv** are either fully validated or were not signed. If validation fails, an explanation of the failure is included in the output; the validation process can be traced in detail. Because **delv** does not rely on an external resolver to carry out validation, it can be used to check the validity of DNS responses in environments where local resolvers may not be trustworthy.

Unless it is told to query a specific name server, **delv** will try each of the servers listed in */etc/resolv.conf*. If no usable server addresses are found, **delv** will send queries to the localhost addresses (127.0.0.1 for IPv4, ::1 for IPv6).

When no command line arguments or options are given, **delv** will perform an NS query for the root zone (.).

4.19.3 Options

@<server>

<server> is the name or IP address of the name server to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied <server> argument is a hostname, **delv** resolves that name before querying that nameserver (note, however, that this initial lookup is **not validated** by DNSSEC).

If this option is not provided, **delv** consults `/etc/resolv.conf`. If an address is found there, it queries the nameserver at that address. If either of the `-4` or `-6` options are in use, then only addresses for the corresponding address family will be tried. If no usable addresses are found, **delv** will send queries to the localhost addresses (`127.0.0.1` for IPv4, `::1` for IPv6).

-a <anchor-file>

Specifies a file from which to read DNSSEC trust anchors. The default is use built-in trust anchors. Keys that do not match the root zone name are ignored. An alternate key name can be specified using the `+root` option.

Note: When reading the trust anchor file, **delv** treats **managed-keys** statements and **trusted-keys** statements identically. That is, for a managed key, it is the *initial* key that is trusted. **RFC 5011** trust anchor management is not supported. **delv** will not consult the managed-keys database maintained by *named(8)*.

-b <address>[#<port>]

Sets the source IP address of the query to <address>. This must be a valid address on one of the host's network interfaces or `0.0.0.0` or `::`. An optional source port may be specified by appending `#<port>`.

-c <class>

Sets the query class for the requested data. Currently, only class *IN* is supported in **delv** and any other value is ignored.

-i

Insecure mode. This disables internal DNSSEC validation. Note, however, this does not set the CD bit on upstream queries. If the server being queried is performing DNSSEC validation, then it will not return invalid data. This can cause **delv** to time out. When it is necessary to examine invalid data to debug a DNSSEC problem, use **dig +cd**.

-p <port>

Specifies a destination port to use for queries. The default is the standard DNS port number 53. This option would be used with a name server that has been configured to listen for queries on a non-standard port number.

-q <name>

Sets the query name to <name>. While the query name can be specified without using the `-q`, it is sometimes necessary to disambiguate names from types or classes (for example, when looking up the name *ns*, which could be misinterpreted as the type NS, or *ch*, which could be misinterpreted as class CH).

-t <type>

Sets the resource record type to query, in the mnemonic form such as *NS* or *AAAA*. It can be any valid query type supported by Loop except for zone transfer types *AXFR* and *IXFR*. As with *-q*, this option is useful to distinguish query name type or class when they are ambiguous. It is sometimes necessary to disambiguate names from types.

The default query type is *A* (IPv4 address), unless the *-x* option is supplied to indicate a reverse lookup, in which case it is *PTR*.

-x <address>

Performs a reverse lookup, mapping an addresses to a name. <address> is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When *-x* is used, there is no need to provide the <name> and <type> arguments. **delv** automatically performs a lookup for a name like *11.12.13.10.in-addr.arpa* and sets the query type to *PTR*. IPv6 addresses are looked up using nibble format under the *IP6.ARPA* domain.

-4

Forces **delv** to only use IPv4.

-6

Forces **delv** to only use IPv6.

-d <level>

Set the debug level to <level>. The allowed range is from 0 to 99. The default is 0 (no debugging). Debugging traces from the program become more verbose as the debug level increases. See the *+mtrace*, *+rtrace*, and *+vtrace* options below for additional debugging details.

-h

Print program usage information and exit.

-v

Print program version and exit.

<name>

The domain name to be looked up.

<type>

Indicates what type of query is required --- *ANY*, *A*, *MX*, etc. <type> can be any valid query type. If this option is not supplied, **delv** will perform a lookup for an *A* (IPv4 address) record.

<class>

Indicates what class of query is required --- *IN*, *CH*, etc. <class> can be any valid query class. If this option is not supplied, **delv** will use class *IN* (Internet).

4.19.4 Query options

delv provides a number of query options which affect the way results are displayed, and in some cases the way lookups are performed.

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string *no* to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form *+keyword=value*. The query options are:

+cdflag

+nocdflag

Controls whether to set the CD (checking disabled) bit in queries sent by **delv**. This may be useful when troubleshooting DNSSEC problems from behind a validating resolver. A validating resolver will block invalid responses, making it difficult to retrieve them for analysis. Setting the CD flag on queries will cause the resolver to return invalid responses, which **delv** can then validate internally and report the errors in detail.

+class

+noclass

Controls whether to display the CLASS when printing a record. The default is to display the CLASS.

+ttl

+nottl

Controls whether to display the TTL when printing a record. The default is to display the TTL.

+rtrace

+nortrace

Toggle resolver fetch logging. This reports the name and type of each query sent by **delv** in the process of carrying out the resolution and validation process: this includes including the original query and all subsequent queries to follow CNAMEs and to establish a chain of trust for DNSSEC validation.

This is equivalent to setting the debug level to 1 in the "resolver" logging category. Setting the debug level to 1 using the *-d* option will produce the same output (but will affect other logging categories as well).

+mtrace

+nomtrace

Toggle message logging. This produces a detailed dump of the responses received by **delv** in the process of carrying out the resolution and validation process.

This is equivalent to setting the debug level to 10 for the "packets" module of the "resolver" logging category. Setting the debug level to 10 using the *-d* option will produce the same output (but will affect other logging categories as well).

+vtrace

+novtrace

Toggle validation logging. This shows the internal process of the validator as it determines whether an answer is validly signed, unsigned, or invalid.

This is equivalent to setting the debug level to 3 for the "validator" module of the "dnssec" logging category. Setting the debug level to 3 using the `:-d` option will produce the same output (but will affect other logging categories as well).

+short

+noshort

Provide a terse answer. The default is to print the answer in a verbose form.

+comments

+nocomments

Toggle the display of comment lines in the output. The default is to print comments.

+rrcomments

+norrrcomments

Toggle the display of per-record comments in the output (for example, human-readable key information about DNSKEY records). The default is to print per-record comments.

+crypto

+nocrypto

Toggle the display of cryptographic fields in DNSSEC records. The contents of these field are unnecessary to debug most DNSSEC validation failures and removing them makes it easier to see the common failures. The default is to display the fields. When omitted they are replaced by the string "[omitted]" or in the DNSKEY case the key id is displayed as the replacement, e.g. "[key id = value]".

+trust

+notrust

Controls whether to display the trust level when printing a record. The default is to display the trust level.

+split=[<W>]

+nosplit

Split long hex- or base64-formatted fields in resource records into chunks of <W> characters (where <W> is rounded up to the nearest multiple of 4). `+nosplit` or `+split=0` causes fields not to be split at all. The default is 56 characters, or 44 characters when multi-line mode is active.

+all

+noall

Set or clear the display options *+comments*, *+rrcomments*, and *+trust* as a group.

+multiline

+nomultiline

Print long records (such as RRSIG, DNSKEY, and SOA records) in a verbose multi-line format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the **delv** output.

+dnssec

+nodnssec

Indicates whether to display RRSIG records in the **delv** output. The default is to do so. Note that (unlike in *dig(1)*) this **does not** control whether to request DNSSEC records or whether to validate them. DNSSEC records are always requested, and validation will always occur unless suppressed by the use of *-i* or *+noroot*.

+root=[<ROOT>]

+noroot

Indicates whether to perform conventional (non-lookaside) DNSSEC validation, and if so, specifies the name of a trust anchor. The default is to validate using a trust anchor of . (the root zone), for which there is a built-in key. If specifying a different trust anchor, then *-a* must be used to specify a file containing the key.

4.19.5 Files

/etc/resolv.conf

4.19.6 See also

dig(1)

4.19.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.20 dig --- DNS client with comprehensive DNS query capabilities

4.20.1 Synopsis

```
dig [@<server>] [-b <address>[#<port>]] [-c <class>] [-f <filename>] [-k <filename>] [-p  
<port>] [-q <name>] [-t <type>] [-x <address>] [-y [<hmac>:]<name>:<key>] [[-4] | [-6]]  
[<name>] [<type>] [<class>] [<query-option> ...]
```

```
dig [<global-query-option> ...] [<query> ...]
```

```
dig [-h | -V]
```

4.20.2 Description

dig is a flexible tool for querying DNS nameservers. It performs DNS lookups and displays the answers that are returned from the nameserver(s) that were queried. DNS administrators may find **dig** useful to troubleshoot DNS problems due to the various options available in the program.

Although **dig** is normally used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command-line arguments and options is printed when the `-h` option is given. **dig** allows multiple lookups to be issued from the command line.

Unless it is told to query a specific nameserver, **dig** will try each of the servers listed in `/etc/resolv.conf`. If no usable server addresses are found, **dig** will send the query to the localhost.

When no command line arguments or options are given, **dig** will perform a type `NS` query for the root name (`.`).

It is possible to set per-user defaults for **dig** via `$HOME/.digrc`. This file is read and any options in it are applied before the command line arguments.

The `IN` and `CH` class names overlap with the `IN` and `CH` top-level domain names. Either use the `-t` and `-c` options to specify the type and class, use the `-q` to specify the domain name, or use `IN.` and `CH.` (with the trailing dot) when looking up these top-level domains.

4.20.3 Options

@<server>

`<server>` is the name or IP address of the nameserver to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied `<server>` argument is a hostname, **dig** resolves that name before querying that nameserver.

If this option is not provided, **dig** consults `/etc/resolv.conf`. If an address is found there, it queries the nameserver at that address. If either of the `-4` or `-6` options are in use, then only addresses for the corresponding address family will

be tried. If no usable addresses are found, **dig** will send queries to the localhost addresses (127.0.0.1 for IPv4, ::1 for IPv6). The reply from the nameserver that responds is displayed.

-4

Forces **dig** to only use IPv4.

-6

Forces **dig** to only use IPv6.

-b <address> [#<port>]

Sets the source IP address of the query to <address>. This must be a valid address on one of the host's network interfaces or 0.0.0.0 or ::. An optional source port may be specified by appending #<port>.

-c <class>

Set the query class. The default class is *IN* (Internet). Other usable classes are *HS* (Hesiod) or *CH* (Chaosnet).

-f <file>

Batch mode. **dig** reads a list of lookup requests to process from the given <file>. Each line in the file should be organized in the same way they would be presented as queries to **dig** using the command-line interface.

-h

Print program usage information and exit.

-i

Do reverse IPv6 lookups using the obsolete [RFC 1886](#) *IP6.INT* domain, which is no longer in use. Obsolete bit string label queries ([RFC 2874](#)) are not attempted.

Error

TODO: Remove this option.

-k <key-file>

Sign queries using TSIG using a key read from the given file. Key files can be generated using *ddns-confgen* (1) **-q**. When using TSIG authentication with **dig**, the nameserver that is queried needs to know the key and algorithm that is being used. In Loop, this is done by providing appropriate **key** and **server** statements in *named.conf* (5).

-p <port>

Specifies a destination port to use for queries. The default is the standard DNS port number 53. This option would be used with a name server that has been configured to listen for queries on a non-standard port number.

-q <name>

Sets the query name to <name>. While the query name can be specified without using the **-q**, it is sometimes necessary to disambiguate names from types

or classes (for example, when looking up the name *ns*, which could be misinterpreted as the type NS, or *ch*, which could be misinterpreted as class CH).

-t <type>

Sets the resource record type to query, in the mnemonic form such as *NS* or *AAAA*. It can be any valid query type supported by Loop. As with *-q*, this option is useful to distinguish query name type or class when they are ambiguous. It is sometimes necessary to disambiguate names from types.

The default query type is *A* (IPv4 address), unless the *-x* option is supplied to indicate a reverse lookup, in which case it is *PTR*. A zone transfer can be requested by specifying a type of *AXFR*. When an incremental zone transfer (IXFR) is required, set the <type> to *ixfr=<N>*. The incremental zone transfer will contain the changes made to the zone since the serial number in the zone's SOA record was <N>.

All resource record types can be expressed in the *TYPE<nn>* mnemonic form, where <nn> is the number of the type. If the resource record type is not supported by Loop, the result will be displayed as described in [RFC 3597](#) for unknown types.

-u

Print query times in microseconds instead of milliseconds.

-x <address>

Performs a reverse lookup, mapping an addresses to a name. <address> is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When *-x* is used, there is no need to provide the <name> and <type> arguments. **dig** automatically performs a lookup for a name like *11.12.13.10.in-addr.arpa* and sets the query type to *PTR*. IPv6 addresses are looked up using nibble format under the IP6.ARPA domain.

-y [<hmac>:]<keyname>:<secret>

Sign queries using TSIG with the authentication key specified directly as an argument. <keyname> is the name of the key, and <secret> is the Base64 encoded shared secret. <hmac> is the name of the key algorithm; valid choices are *hmac-md5*, *hmac-sha1*, *hmac-sha224*, *hmac-sha256*, *hmac-sha384*, or *hmac-sha512*. If <hmac> is not specified, the default is *hmac-sha256*.

Warning

You should use the *-k* option and avoid the *-y* option, because with *-y* the shared secret is supplied as a command line argument in clear text. This may be visible in the history file maintained by the user's shell and by other means.

Error

TODO: Check the supported HMAC algorithms.

-v

Print program version and exit.

<name>

The domain name to be looked up.

<type>

Indicates what type of query is required --- *ANY*, *A*, *MX*, etc. **<type>** can be any valid query type. If this option is not supplied, **dig** will perform a lookup for an *A* (IPv4 address) record.

<class>

Indicates what class of query is required --- *IN*, *CH*, etc. **<class>** can be any valid query class. If this option is not supplied, **dig** will use class *IN* (Internet).

4.20.4 Query options

dig provides a number of query options which affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies.

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string *no* to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form *+keyword=value*. Keywords may be abbreviated, provided the abbreviation is unambiguous; for example, **+cd** is equivalent to **+cdf***flag*. The query options are:

+aaflag

+noaaflag

Controls whether to set the **AA** DNS message header flag in the query.

+additional

+noadditional

Controls whether to display the additional section of a reply DNS message. The default is to display it.

+adflag

+noadflag

Controls whether to set the **AD** DNS message header flag in the query. This flag requests the server to return whether all of the answer and authority sections have all been validated as secure according to the security policy of the server. *AD=1* indicates that all records have been validated as secure and the answer is not from an NSEC3 Opt-Out range. *AD=0* indicate that some part of the answer was insecure or not validated. This bit is set by default.

+all

+noall

Set or clear all display options.

+answer

+noanswer

Controls whether to display the answer section of a reply DNS message. The default is to display it.

+authority

+noauthority

Controls whether to display the authority section of a reply DNS message. The default is to display it.

+besteffort

Attempt to display the contents of messages which are malformed.

+nobesteffort

Do not display the contents of messages which are malformed. This is the default.

+bufsize=<udp-buffer-size>

Set the UDP message buffer size advertised using EDNS0 to *<udp-buffer-size>* bytes. The minimum and maximum sizes of this buffer are 0 and 65535 bytes respectively. Values outside this range are rounded up or down appropriately. Values other than zero will cause a EDNS query to be sent.

+cdflag

+nocdflag

Controls whether to set the **CD** DNS message header flag in the query. This flag requests the server to not perform DNSSEC validation of responses.

+class

+noclass

Controls whether to display the CLASS when printing the record. The default is to display the class.

+cmd

+nocmd

Controls the printing of an initial comment in the output identifying the version of **dig** and the query options that have been applied. This comment is printed by default.

+comments

+nocomments

Controls the the display of comment lines in the output. The default is to print comments.

+cookie=[<value>]

+nocookie

Send an COOKIE EDNS option, containing an optional <value>. Replaying a COOKIE from a previous response will allow the server to identify a previous client. The default is not to send cookies.

+cookie is automatically set when *+trace* is in use, to better emulate the default queries from a nameserver.

+crypto

+nocrypto

Controls the display of cryptographic fields in DNSSEC records. The contents of these fields are unnecessary to debug most DNSSEC validation failures and removing them makes it easier to see the common failures. The default is to display the fields. When omitted they are replaced by the string *[omitted]* or in the DNSKEY case the key id is displayed as the replacement, e.g. *[key id = value]*.

+dnssec

+nodnssec

Requests DNSSEC records be sent by setting the **DO** (DNSSEC OK) bit in the OPT record in the additional section of the query.

+domain=<somename>

Set the search list to contain the single domain <somename>, as if specified in a **domain** directive in */etc/resolv.conf*, and enable search list processing as if the *+search* option were given.

+edns=[<version>]

+noedns

Specify the EDNS version to query with. Valid values for <version> are 0 to 255. Setting the EDNS version will cause a EDNS query to be sent. *+noedns* clears the remembered EDNS version. By default, EDNS is set to 0.

+ednsflags=[<flags>]

+noednsflags

Set the must-be-zero EDNS flags bits (**Z** bits) to the specified value. Decimal, hex and octal encodings are accepted. Setting a named flag (e.g. **DO**) will silently be ignored. By default, no **Z** bits are set.

+ednsnegotiation

+noednsnegotiation

Controls EDNS version negotiation. By default, EDNS version negotiation is enabled.

+ednsopt=<code>[:<value>]

+noednsopt

Specify EDNS option with code point *<code>* and optionally payload of *<value>* as a hexadecimal string. *<code>* can be either an EDNS option name (for example, *NSID* or *ECS*), or an arbitrary numeric value. *+noednsopt* clears the EDNS options to be sent.

+expire**+noexpire**

Controls sending of an EDNS Expire option ([RFC 7314](#)).

+fail**+nofail**

Do not try the next server if you receive a SERVFAIL. The default is to not try the next server which is the reverse of normal stub resolver behavior.

+identify**+noidentify**

Controls display of the IP address and port number that supplied the answer when the *+short* option is enabled. If short form answers are requested, the default is not to show the source address and port number of the server that provided the answer.

+ignore**+noignore**

Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+keepopen**+nokeepopen**

Keep the TCP socket open between queries and reuse it rather than creating a new TCP socket for each lookup. The default is to keep the TCP socket open.

+multiline**+nomultiline**

Print records like the SOA records in a verbose multi-line format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the **dig** output.

+ndots=<num-dots>

Set the number of dots that have to appear in *<name>* to *<num-dots>* for it to be considered absolute. The default value is that defined using the **ndots** statement in */etc/resolv.conf*, or *1* if no **ndots** statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the **search** or **domain** directive in */etc/resolv.conf* if *+search* is set.

+nsid

+nonsid

Include an EDNS nameserver ID request option ([RFC 5001](#)) when sending a query.

+nssearch

+nonssearch

When this option is set, **dig** attempts to find the authoritative name servers for the zone containing the name being looked up and display the SOA record that each nameserver has for the zone.

+onesoa

+noonesoa

Print only one (starting) SOA record when performing an AXFR. The default is to print both the starting and ending SOA records.

+opcode=<value>

+noopcode

Set/restore the DNS message opcode to the specified value. The default value is 0 for DNS QUERY.

+qr

+noqr

Print / do not print the query as it is sent. By default, the query is not printed.

+question

+noquestion

Controls whether to display the question section of a reply DNS message. The default is to display it as a comment.

+rdflag

+nordflag

Toggle the setting of the **RD** DNS message header flag in the query. This bit is set by default, which means **dig** normally sends recursive queries. Recursion is automatically disabled when the *+nssearch* or *+trace* query options are used.

+recurse

+norecurse

A synonym for *+no[no]rdflag*.

Error

TODO: Remove this option.

+retry=<count>

Sets the number of times to retry UDP queries to server to *<count>*. The default is 2. Unlike *+tries*, this does not include the initial query.

+rrcomments

+norrcomments

Toggle the display of per-record comments in the output (for example, human-readable key information about DNSKEY records). The default is not to print record comments unless multiline mode is active.

+search

+nosearch

Use / do not use the search list defined by the **searchlist** or **domain** directive in */etc/resolv.conf* (if any). The search list is not used by default.

ndots from */etc/resolv.conf* which may be overridden by *+ndots* determines if the name will be treated as relative or not and hence whether a search is eventually performed or not.

+short

+noshort

Provide a terse answer. The default is to print the answer in a verbose form.

+showsearch

+noshowsearch

Perform / do not perform a search showing intermediate results.

+sit

+nosit

This option is a synonym for *+no(cookie)*. The *+no(sit)* is deprecated.

Error

TODO: Remove this option.

+split=<num-chars>

+nosplit

Split long hex- or base64-formatted fields in resource records into chunks of *<num-chars>* characters (where *<num-chars>* is rounded up to the nearest multiple of 4). *+nosplit* or *+split=0* causes fields not to be split at all. The default is 56 characters, or 44 characters when multiline mode is active.

+stats

+nostats

This query option toggles the printing of statistics: when the query was made, the size of the reply and so on. The default behavior is to print the query statistics.

+subnet=<address>[/<prefix-length>]

+nosubnet

Controls sending of an EDNS Client Subnet option ([RFC 7871](#)) with the specified network prefix.

Using `+subnet=0.0.0.0/0`, or simply `+subnet=0` for short, sends an EDNS CLIENT-SUBNET option with an empty address and a source prefix-length of zero, which signals to a resolver that the client's address information **must not** be used when resolving this query.

+tcp

+notcp

Use / do not use TCP when querying nameservers. The default behavior is to use UDP unless a type *ixfr*=<N> query is requested, in which case the default is TCP. AXFR queries always use TCP.

+time=<timeout>

Sets the timeout for a query to <timeout> seconds. The default timeout is 5 seconds. If <timeout> is less than 1, it is silently set to 1.

+trace

+notrace

Toggle tracing of the delegation path from the root nameservers for the name being looked up. Tracing is disabled by default. When tracing is enabled, **dig** makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

If @<server> is also specified, it affects only the initial query for the root zone nameservers.

`+dnssec` is also set when `+trace` is set to better emulate the default queries from a nameserver.

+tries=<count>

Sets the number of times to try UDP queries to server to <count>. The default is 3. If <count> is less than 1, it is silently set to 1.

+ttlid

+nottlid

Display / do not display the TTL when printing the record.

ErrorTODO: Rename this to **+ttl**.**+vc****+novc**This is an alias for **+tcp**. The "vc" stands for "virtual circuit".**Error**

TODO: Remove this option.

4.20.5 Multiple queries

dig supports specifying multiple queries on the command line, in addition to supporting the **-f** option. Each of those queries can be supplied with its own set of flags, options and query options.

In this case, each *<query>* argument represent an individual query in the command-line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type and class and any query options that should be applied to that query.

A global set of query options, which should be applied to all queries, can also be supplied as *<global-query-option>*. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the **+cmd** and **+nocmd** options) can be overridden by a query-specific set of query options. For example:

```
$ dig +qr www.banu.com any -x 127.0.0.1 banu.com ns +noqr
```

shows how **dig** could be used from the command line to make three lookups: a type *ANY* query for *www.banu.com*, a reverse lookup of *127.0.0.1* and a query for the type *NS* records of *banu.com*. A global query option **+qr** is applied, so that **dig** shows the initial query it made for each lookup. The final query has a local query option of **+noqr** which means that **dig** will not print the initial query when it looks up the *NS* records for *banu.com*.

4.20.6 Files

`/etc/resolv.conf``$HOME/.digrc`

4.20.7 See also

delv(1), *host(1)*, *named(8)*, *dnssec-keygen(1)*, *ddns-confgen(1)*

4.20.8 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2011, 2013-2018 Internet Systems Consortium, Inc. ("ISC").

4.21 host - Simple DNS lookup client

4.21.1 Synopsis

host [-a] [-C] [-d] [-l] [-n] [-r] [-s] [-T] [-w] [-v] [-c <class>] [-N <num-dots>] [-R <number>] [-t <type>] [-W <num-seconds>] [-m <flag>] [[-4] | [-6]] [-v] <name> [<server>]

host [-h | -V]

4.21.2 Description

host is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa. When no arguments or options are given, **host** prints a short summary of its command line arguments and options.

<name> is the domain name that is to be looked up. It can also be a dotted-decimal IPv4 address or a colon-delimited IPv6 address, in which case **host** will by default perform a reverse lookup for that address. <server> is an optional argument which is either the name or IP address of the nameserver that **host** should query instead of the server or servers listed in `/etc/resolv.conf`.

Warning

This program may be removed from the Loop distribution in the future. Please use *dig(1)* instead.

4.21.3 Options

-4

Use IPv4 only for query transport. See also the **-6** option.

-6

Use IPv6 only for query transport. See also the **-4** option.

-a

Stands for *all*. The **-a** option is normally equivalent to **-v -tANY**. It also affects the behaviour of the **-l** list zone option.

-c <class>

Set the query class. The default class is *IN* (Internet). Other usable classes are *HS* (Hesiod) or *CH* (Chaosnet).

-C

Check consistency. **host** will query the SOA records for zone <name> from all the listed authoritative nameservers for that zone. The list of nameservers is defined by the NS records that are found for the zone.

-d

Print debugging traces. Equivalent to the **-v** option.

Error

TODO: Combine **-d** and **-v**.

-h

Print program usage information and exit.

-i

Do reverse IPv6 lookups using the obsolete **RFC 1886** *IP6.INT* domain, which is no longer in use. Obsolete bit string label queries (**RFC 2874**) are not attempted.

Error

TODO: Remove this option.

-l

List zone. The **host** command performs a zone transfer of zone <name> and prints out the NS, PTR and address records (A/AAAA).

Together, the **-l -a** options print all records in the zone.

-N <num-ndots>

The number of dots that have to be in <name> for it to be considered absolute. The default value is that defined using the **ndots** statement in */etc/resolv.conf*, or *1* if no **ndots** statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the **search** or **domain** directive in */etc/resolv.conf*.

-r

Non-recursive query. Setting this option clears the **RD** DNS message header flag in the query. This should mean that the nameserver receiving the query will not attempt to resolve <name>. The **-r** option enables **host** to mimic the behavior of a nameserver by making non-recursive queries and expecting to receive answers to those queries that can be referrals to other nameservers.

-R <count>

Number of retries for UDP queries. If <count> is less than 1, it will be set to 1. The default value is 1.

-s

Do not send the query to the next nameserver if any server responds with a SERVFAIL response, which is the reverse of normal stub resolver behavior.

-t <type>

Specifies the record type. The <type> argument can be any recognized RR type: CNAME, NS, SOA, TXT, DNSKEY, AXFR, etc.

When no query type is specified, **host** automatically selects an appropriate query type. By default, it looks for A, AAAA, and MX records. If the **-C** option is given, queries will be made for SOA records. If <name> is a dotted-decimal IPv4 address or colon-delimited IPv6 address, **host** will query for PTR records.

If a query type of IXFR is chosen the starting serial number can be specified by appending an equal followed by the starting serial number (like **-t IXFR=12345678**).

-T

Use TCP. By default, **host** uses UDP when making queries. The **-T** option makes it use a TCP connection when querying the nameserver. TCP will be automatically selected for queries that require it, such as zone transfer (AXFR) requests.

-v

Print debugging traces. Equivalent to the **-d** option.

Error

TODO: Combine **-d** and **-v**.

-V

Print program version and exit.

-w

Wait forever: The query timeout is set to the maximum possible. See also the **-W** option.

-W <num-seconds>

Timeout: Wait for up to <num-seconds> seconds for a reply. If <num-seconds> is less than 1, the wait interval is set to 1 second.

By default, **host** will wait for 5 seconds for UDP responses and 10 seconds for TCP connections.

Also see the **-w** option.

4.21.4 Files

`/etc/resolv.conf`

4.21.5 See also

dig(1), *named(8)*

4.21.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2002, 2004-2005, 2007-2009, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.22 nsupdate - DNS client to send dynamic DNS UPDATES to a nameserver

4.22.1 Synopsis

nsupdate [-d] [-D] [-i] [-L level] [[-g] | [-o] | [-l] | [-y [hmac:]keyname:secret] | [-k keyfile]] [-t timeout] [-u <udp-timeout>] [-r <udp-retries>] [-v] [-T] [-P] [filename]

nsupdate [-h | -V]

4.22.2 Description

nsupdate is used to submit DNS UPDATE requests ([RFC 2136](#)) to a nameserver. DNS UPDATE allows resource records to be added or removed from a zone dynamically without manually editing the zone file. A single DNS UPDATE request can contain requests to add or remove more than one resource record.

Master files of zones that are under dynamic control via **nsupdate** should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with **nsupdate** have to be in the same zone. Requests are sent to the zone's master server. This is identified by the MNAME field of the zone's SOA record.

Transaction signatures can be used to authenticate the DNS UPDATE messages. These use TSIG signatures ([RFC 2845](#)) or SIG(0) signatures ([RFC 2535](#) and [RFC 2931](#)).

TSIG relies on a shared secret that should only be known to **nsupdate** and the name server. For instance, suitable **key** and **server** statements would be added to *named.conf(5)* so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will be using TSIG authentication. You can use *ddns-confgen(1)* to generate suitable configuration fragments. **nsupdate** uses the *-y* or *-k* options to provide the TSIG shared secret. These options are mutually exclusive.

SIG(0) uses public key cryptography. To use a SIG(0) key, the public key must be stored in a KEY record in a zone served by the name server.

GSS-TSIG uses Kerberos credentials. Standard GSS-TSIG mode is switched on with the `-g` option. A non-standards-compliant variant of GSS-TSIG used by Windows 2000 can be switched on with the `-o` option.

4.22.3 Options

- d**
Debug mode. This provides tracing information about the update requests that are made and the replies received from the name server.
- D**
Extra debug mode.
- h**
Print program usage information and exit.
- i**
Force interactive mode, even when standard input is not a terminal.
- k** <keyfile>
The file containing the TSIG authentication key. Keyfiles may be in two formats: a single file containing a *named.conf*(5) format **key** statement, which may be generated automatically by *ddns-confgen*(1), or a pair of files whose names are of the format *Kname*.+157.+*random*.key and *Kname*.+157.+*random*.private, which can be generated by *dnssec-keygen*(1). The **-k** may also be used to specify a SIG(0) key used to authenticate Dynamic DNS update requests. In this case, the key specified is not an HMAC-MD5 key.
- l**
Local-host only mode. This sets the server address to localhost (disabling the **server** so that the server address cannot be overridden). Connections to the local server will use a TSIG key found in */var/run/loop/session.key*, which is automatically generated by *named*(8) if any local master zone has set **update-policy** to *local*. The location of this key file can be overridden with the **-k** option.
- L** <level>
Set the logging debug level. If <level> is 0, logging is disabled.
- p** <port>
Set the port to use for connections to a name server. The default is 53.
- P**
Print the list of private Loop-specific resource record types whose format is understood by **nsupdate**. See also the **-T** option.
- r** <udp-retries>
The number of UDP retries. The default is 3. If <udp-retries> is 0, only 1 update request will be made.

-t <timeout>

The maximum time an update request can take before it is aborted. The default is 300 seconds. 0 can be used to disable the timeout.

-T

Print the list of IANA standard resource record types whose format is understood by **nsupdate**. **nsupdate** will exit after the lists are printed. The **-T** option can be combined with the **-P** option.

Other types can be entered using the "TYPE<XXXXX>" syntax where "XXXXX" is the decimal value of the type with no leading zeros. The rdata, if present, will be parsed using the unknown RDATA format ([RFC 3597](#)).

-u <udp-timeout>

The UDP retry interval. The default is 3 seconds. If 0 is passed, the interval will be computed from the timeout interval and number of UDP retries.

-v

Use TCP even for small update requests. By default, **nsupdate** uses UDP to send update requests to the name server unless they are too large to fit in a UDP request in which case TCP will be used. TCP may be preferable when a batch of update requests is made.

-V

Print program version and exit.

-y [<hmac>:]<keyname>:<secret>

Sign queries using TSIG with the authentication key specified directly as an argument. <keyname> is the name of the key, and <secret> is the Base64 encoded shared secret. <hmac> is the name of the key algorithm; valid choices are *hmac-md5*, *hmac-sha1*, *hmac-sha224*, *hmac-sha256*, *hmac-sha384*, or *hmac-sha512*. If <hmac> is not specified, the default is *hmac-sha256*.

Warning

You should use the **-k** option and avoid the **-y** option, because with **-y** the shared secret is supplied as a command line argument in clear text. This may be visible in the history file maintained by the user's shell and by other means.

Error

TODO: Check the supported HMAC algorithms.

4.22.4 Input commands

nsupdate reads input from *<filename>* or standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates will be rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This allows a suitably authenticated update request to proceed if some specified resource records are present or missing from the zone. A blank input line (or the *send* command) causes the accumulated commands to be sent as one DNS UPDATE request to the name server.

Lines beginning with a semicolon are comments and are ignored.

The command formats and their meaning are as follows:

server *<servername>* [*<port>*]

Sends all dynamic update requests to the name server *<servername>*. When no server statement is provided, **nsupdate** will send updates to the master server of the correct zone. The MNAME field of that zone's SOA record will identify the master server for that zone. *<port>* is the port number on *<servername>* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

local *<address>* [*<port>*]

Sends all dynamic update requests using the local *<address>*. When no *local* statement is provided, **nsupdate** will send updates using an address and port chosen by the system. *<port>* can additionally be used to make requests come from a specific port. If no port number is specified, the system will assign one.

zone *<zonenumber>*

Specifies that all updates are to be made to the zone *<zonenumber>*. If no *zone* statement is provided, **nsupdate** will attempt determine the correct zone to update based on the rest of the input.

class *<classname>*

Specify the default class. If no *<class>* is specified, the default class is *IN*.

ttd *<seconds>*

Specify the default TTL for records to be added. The value *none* will clear the default TTL.

key [*<hmac>*:] *<keyname>* *<secret>*

Specifies that all updates are to be TSIG-signed using the *<keyname>* *<secret>* pair. If *<hmac>* is specified, then it sets the signing algorithm to use. The default is *hmac-sha256*. The *key* statement overrides any key specified on the command line via *-y* or *-k*.

gsstsig

Use GSS-TSIG to sign the updates. This is equivalent to specifying *-g* on the commandline.

oldgsstsig

Use the Windows 2000 version of GSS-TSIG to sign the updates. This is equivalent to specifying *-o* on the commandline.

realm [*<realm-name>*]

When using GSS-TSIG, use *<realm-name>* rather than the default realm in *krb5.conf*. If no realm name is specified, the saved realm is cleared.

prereq *nxdomain* *<owner-name>*

Requires that no resource record of any type exists with name *<owner-name>*.

prereq *yxdomain* *<owner-name>*

Requires that *<owner-name>* exists (has as at least one resource record, of any type).

prereq *nxrrset* *<owner-name>* [*<class>*] *<type>*

Requires that a resource record **does not** exist of the specified *<type>*, *<class>* and *<owner-name>*. If *<class>* is omitted, *IN* is assumed.

prereq *yxrrset* *<owner-name>* [*<class>*] *<type>* [*<rdata>*]

If *<rdata>* is not specified, this statement requires that a resource record **does** exist of the specified *<type>*, *<class>* and *<owner-name>*. If *<class>* is omitted, *IN* is assumed.

If *<rdata>* is specified, the *<data>* field from each set of prerequisites of this form sharing a common *<type>*, *<class>*, and *<owner-name>* are combined to form an RRSets. This RRSets must exactly match the RRSets existing in the zone for the given *<type>*, *<class>*, and *<owner-name>*. The *<rdata>* fields are written in the presentation format of the resource record's RDATA.

update *del[ete]* *<owner-name>* [*<ttl>*] [*<class>*] [*<type>*] [*<rdata>*]

Deletes any resource records named *<owner-name>*. If *<type>* and/or *<rdata>* are provided, only matching resource records will be removed. If *<class>* is not supplied, it defaults to *IN*. The *<rdata>* fields are written in the presentation format of the resource record's RDATA. The *<ttl>* argument is ignored, and is only allowed for compatibility.

update *add* *<owner-name>* *<ttl>* [*<class>*] *<type>* [*<rdata>*]

Adds a new resource record with the specified *<owner-name>*, *<ttl>*, *<class>* and *<rdata>*. If *<class>* is not supplied, it defaults to *IN*. The *<rdata>* fields are written in the presentation format of the resource record's RDATA.

show

Displays the current message, containing all of the prerequisites and updates specified since the last send.

send

Sends the current message. This is equivalent to entering a blank line.

answer

Displays the answer.

debug

Turn on debugging.

version

Print the program's version number.

help

Print a list of supported commands.

4.22.5 Examples

The examples below show how **nsupdate** could be used to insert and delete resource records from the *example.com* zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for *example.com*.

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
> send
```

Any A records for *oldhost.example.com* are deleted. And an A record for *newhost.example.com* with IP address 172.16.1.1 is added. The newly-added record has a 1 day TTL (86400 seconds).

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.
  ↪example.com
> send
```

The prerequisite condition gets the name server to check that there are no resource records of any type for *nickname.example.com*. If there are, the update request fails. If this name does not exist, a CNAME for it is added. This ensures that when the CNAME is added, it cannot conflict with the long-standing rule in [RFC 1034](#) that a name must not exist as any other record type if it exists as a CNAME. (The rule has been updated for DNSSEC in [RFC 2535](#) to allow CNAMEs to have RRSIG, DNSKEY and NSEC records.)

4.22.6 Files

`/etc/resolv.conf`:: used to identify default name server

`/var/run/loop/session.key`:: sets the default TSIG key for use in local-only mode

`Kname.+157.+random.key`:: Base64 encoding of HMAC-MD5 key created by `dnssec-keygen(8)`.

`Kname.+157.+random.private`:: Base64 encoding of HMAC-MD5 key created by `dnssec-keygen(8)`.

The TSIG key is redundantly stored in two separate files. This is a consequence of `nsupdate` using the DST library for its cryptographic operations, and may change in future releases.

Error

TODO: Some of the description about files needs to be revisited.

4.22.7 See also

`ddns-confgen(1)`, `named.conf(5)`

4.22.8 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2000-2012, 2014-2018 Internet Systems Consortium, Inc. ("ISC").

4.23 `dnssperf` --- DNS client that measures DNS name-server performance

4.23.1 Synopsis

dnssperf [-a local_addr] [-b bufsize] [-c clients] [-d datafile] [-D] [-e] [-f family] [-l limit] [-n runs_through_file] [-p port] [-q num_queries] [-Q max_qps] [-s server_addr] [-S stats_interval] [-t timeout] [-T threads] [-u] [-v] [-x local_port] [-y [alg:]name:secret]

dnssperf [-h | -V]

4.23.2 Description

dnssperf is a DNS server performance testing tool. It is primarily intended for measuring the performance of authoritative DNS servers, but it can also be used for measuring caching server performance in a closed laboratory environment. For testing caching servers resolving against the live internet, the `resperf(1)` program is preferred.

It is recommended that `dnssperf` and the name server under test be run on separate machines, so that the CPU usage of `dnssperf` itself does not slow down the name server. The two machines should be connected with a fast network, preferably a dedicated Gigabit Ethernet segment. Testing through a router or firewall is not advisable.

4.23.2.1 Configuring the name server

If using `dnssperf` to test an authoritative server, the name server under test should be set up to serve one or more zones similar in size and number to what the server is expected to serve in production.

Also, be sure to turn off recursion in the server's configuration (specify "recursion no;" in the options block).

4.23.2.2 Constructing a query input file

A `dnssperf` input file should contain a large and realistic set of queries, on the order of ten thousand to a million. The input file contains one line per query, consisting of a domain name and an RR type name separated by a space. The class of the query is implicitly IN.

When measuring the performance serving non-terminal zones such as the root zone or TLDs, note that such servers spend most of their time providing referral responses, not authoritative answers. Therefore, a realistic input file might consist mostly of queries for type A for names *below*, not at, the delegations present in the zone. For example, when testing the performance of a server configured to be authoritative for the top-level domain "fi.", which contains delegations for domains like "helsinki.fi" and "turku.fi", the input file could contain lines like:

```
www.turku.fi A
www.helsinki.fi A
```

where the "www" prefix ensures that the server will respond with a referral. Ideally, a realistic proportion of queries for nonexistent domains should be mixed in with those for existing ones, and the lines of the input file should be in a random order.

4.23.2.3 Constructing a dynamic update input file

To test dynamic update performance, `dnssperf` is run with the `-u` option, and the input file is constructed of blocks of lines describing dynamic update messages. The first line in a block contains the zone name:

```
example.com
```

Subsequent lines contain prerequisites, if there are any. Prerequisites can specify that a name may or may not exist, an rrset may or may not exist, or an rrset exists and its rdata matches all specified rdata for that name and type. The keywords "require" and "prohibit" are followed by the appropriate information. All relative names are considered to be relative to the zone name. The following lines show the 5 types of prerequisites:

```
require a
require a A
require a A 1.2.3.4
prohibit x
prohibit x A
```

Subsequent lines contain records to be added, records to be deleted, rrsets to be deleted, or names to be deleted. The keywords "add" or "delete" are followed by the appropriate information. All relative names are considered to be relative to the zone name. The following lines show the 4 types of updates:

```
add x 3600 A 10.1.2.3
delete y A 10.1.2.3
delete z A
delete w
```

Each update message is terminated by a line containing the command:

```
send
```

4.23.2.4 Running the tests

When running **dnssperf**, a data file (the -d option) and server (the -s option) will normally be specified. The output of dnssperf is mostly self-explanatory. Pay attention to the number of dropped packets reported - when running the test over a local Ethernet connection, it should be zero. If one or more packets has been dropped, there may be a problem with the network connection. In that case, the results should be considered suspect and the test repeated.

4.23.3 Options

-a <local_addr>

Specifies the local address from which to send requests. The default is the wild-card address.

-b <bufsize>

Sets the size of the socket's send and receive buffers, in kilobytes. If not specified, the operating system's default is used.

-c clients

Act as multiple clients. Requests are sent from multiple sockets. The default is to act as 1 client.

-d datafile

Specifies the input data file. If not specified, dnssperf will read from standard input.

- D**
Sets the DO (DNSSEC OK) bit [RFC3225] in all packets sent. This also enables EDNS0, which is required for DNSSEC.
- e**
Enables EDNS0 [RFC2671], by adding an OPT record to all packets sent.
- f** family
Specifies the address family used for sending DNS packets. The possible values are "inet", "inet6", or "any". If "any" (the default value) is specified, dnssperf will use whichever address family is appropriate for the server it is sending packets to.
- h**
Print program usage information and exit.
- l** limit
Specifies a time limit for the run, in seconds. This may cause the input to be read multiple times, or only some of the input to be read. The default behavior is to read the input once, and have no specific time limit.
- n** runs_through_file
Run through the input file at most this many times. If no time limit is set, the file will be read exactly this number of times; if a time limit is set, the file may be read fewer times.
- p** port
Sets the port on which the DNS packets are sent. If not specified, the standard DNS port (53) is used.
- q** num_queries
Sets the maximum number of outstanding requests. When this value is reached, dnssperf will not send any more requests until either responses are received or requests time out. The default value is 100.
- Q** max_qps
Limits the number of requests per second. There is no default limit.
- s** server_addr
Specifies the name or address of the server to which requests will be sent. The default is the loopback address, 127.0.0.1.
- S** stats_interval
If this parameter is specified, a count of the number of queries per second during the interval will be printed out every stats_interval seconds.
- t** timeout
Specifies the request timeout value, in seconds. dnssperf will no longer wait for a response to a particular request after this many seconds have elapsed. The default is 5 seconds.

-T `threads`

Run multiple client threads. By default, `dnssperf` uses one thread for sending requests and one thread for receiving responses. If this option is specified, `dnssperf` will instead use `N` pairs of send/receive threads.

-u

Instructs `dnssperf` to send DNS dynamic update messages, rather than queries. The format of the input file is different in this case; see the "Constructing a dynamic update input file" section for more details.

-v

Enables verbose mode. The DNS RCODE of each response will be reported to standard output when the response is received, as will the latency. If a query times out, it will be reported with the special string "T" instead of a normal DNS RCODE. If a query is interrupted, it will be reported with the special string "I".

-V

Print program version and exit.

-x `local_port`

Specifies the local port from which to send requests. The default is the wildcard port (0).

If acting as multiple clients and the wildcard port is used, each client will use a different random port. If a port is specified, the clients will use a range of ports starting with the specified one.

-y [`alg:`] `name:secret`

Add a TSIG record [RFC2845] to all packets sent, using the specified TSIG key algorithm, name and secret, where the algorithm defaults to `hmac-md5` and the secret is expressed as a base-64 encoded string.

4.23.4 See also

resperf(1)

4.23.5 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (C) 2000, 2001, 2004-2015 Nominum, Inc.

4.24 `resperf` --- DNS client that measures DNS resolver performance

4.24.1 Synopsis

resperf [`-a` `local_addr`] [`-d` `datafile`] [`-s` `server_addr`] [`-p` `port`] [`-x` `local_port`] [`-t` `timeout`] [`-b` `bufsize`] [`-f` `family`] [`-e`] [`-D`] [`-y` [`alg:`] `name:secret`] [`-i` `interval`] [`-m` `max_qps`] [`-P`

plot_data_file] [-r rampup_time] [-c constant_traffic_time] [-L max_loss] [-C clients] [-q max_outstanding]

resperf [-h | -V]

resperf-report [-a local_addr] [-d datafile] [-s server_addr] [-p port] [-x local_port] [-t timeout] [-b bufsize] [-f family] [-e] [-D] [-y [alg:]name:secret] [-i interval] [-m max_qps] [-r rampup_time] [-c constant_traffic_time] [-L max_loss] [-C clients] [-q max_outstanding]

resperf-report [-h | -V]

4.24.2 Description

resperf is a companion tool to *dnssperf(1)*. *dnssperf* was primarily designed for benchmarking authoritative servers, and it does not work well with caching servers that are talking to the live Internet. One reason for this is that *dnssperf* uses a "self-pacing" approach, which is based on the assumption that you can keep the server 100% busy simply by sending it a small burst of back-to-back queries to fill up network buffers, and then send a new query whenever you get a response back. This approach works well for authoritative servers that process queries in order and one at a time; it also works pretty well for a caching server in a closed laboratory environment talking to a simulated Internet that's all on the same LAN. Unfortunately, it does not work well with a caching server talking to the actual Internet, which may need to work on thousands of queries in parallel to achieve its maximum throughput. There have been numerous attempts to use *dnssperf* (or its predecessor, *queryperf*) for benchmarking live caching servers, usually with poor results. Therefore, a separate tool designed specifically for caching servers is needed.

4.24.2.1 How resperf works

Unlike the "self-pacing" approach of *dnssperf*, *resperf* works by sending DNS queries at a controlled, steadily increasing rate. By default, *resperf* will send traffic for 60 seconds, linearly increasing the amount of traffic from zero to 100,000 queries per second.

During the test, *resperf* listens for responses from the server and keeps track of response rates, failure rates, and latencies. It will also continue listening for responses for an additional 40 seconds after it has stopped sending traffic, so that there is time for the server to respond to the last queries sent. This time period was chosen to be longer than the overall query timeout of Loop.

If the test is successful, the query rate will at some point exceed the capacity of the server and queries will be dropped, causing the response rate to stop growing or even decrease as the query rate increases.

The result of the test is a set of measurements of the query rate, response rate, failure response rate, and average query latency as functions of time.

4.24.2.2 What you will need

Benchmarking a live caching server is serious business. A fast caching server running on a powerful machine, resolving a mix of cacheable and non-cacheable queries typical of ISP customer traffic, is capable of resolving over 100,000 queries per second. In the process, it will send more than 40,000 queries per second to authoritative servers on the Internet, and receive responses to most of them. Assuming an average request size of 50 bytes and a response size of 150 bytes, this amounts to some 16 Mbps of outgoing and 48 Mbps of incoming traffic. If your internet connection can't handle the bandwidth, you will end up measuring the speed of the connection, not the server, and may saturate the connection causing a degradation in service for other users.

Make sure there is no stateful firewall between the server and the Internet, because most of them can't handle the amount of UDP traffic the test will generate and will end up dropping packets, skewing the test results. Some will even lock up or crash.

You should run `resperf` on a machine separate from the server under test, on the same LAN. Preferably, this should be a Gigabit Ethernet network. The machine running `resperf` should be at least as fast as the machine being tested; otherwise, it may end up being the bottleneck.

There should be no other applications running on the machine running `resperf`. Performance testing at the traffic levels involved is essentially a hard real-time application - consider the fact that at a query rate of 100,000 queries per second, if `resperf` gets delayed by just 1/100 of a second, 1000 incoming UDP packets will arrive in the meantime. This is more than most operating systems will buffer, which means packets will be dropped.

Because the granularity of the timers provided by operating systems is typically too coarse to accurately schedule packet transmissions at sub-millisecond intervals, `resperf` will busy-wait between packet transmissions, constantly polling for responses in the meantime. Therefore, it is normal for `resperf` to consume 100% CPU during the whole test run, even during periods where query rates are relatively low.

You will also need a set of test queries in the `dnssperf` file format. See the `dnssperf` man page for instructions on how to construct this query file. To make the test as realistic as possible, the queries should be derived from recorded production client DNS traffic, without removing duplicate queries or other filtering. With the default settings, `resperf` will use up to 3 million queries in each test run.

If the caching server to be tested has a configurable limit on the number of simultaneous resolutions, like the `recursive-clients` option in `named.conf(5)`, you will probably have to increase it. As a starting point, we recommend a value of 100000 for Loop. Should the limit be reached, it will show up in the plots as an increase in the number of failure responses.

The server being tested should be restarted at the beginning of each test to make sure it is starting with an empty cache. If the cache already contains data from a previous test run that used the same set of queries, almost all queries will be answered from the cache, yielding inflated performance numbers.

To use the **`resperf-report`** script, you need to have `gnuplot` installed. Make sure

your installed version of gnuplot supports the png terminal driver. If your gnuplot doesn't support png but does support gif, you can change the line saying `terminal=png` in the `resperf-report` script to `terminal=gif`.

4.24.2.3 Running the test

resperf is typically invoked via the `resperf-report` script, which will run `resperf` with its output redirected to a file and then automatically generate an illustrated report in HTML format. Command line arguments given to `resperf-report` will be passed on unchanged to `resperf`.

When running `resperf-report`, you will need to specify at least the server IP address and the query data file. A typical invocation will look like:

```
resperf-report -s 10.0.0.2 -d queryfile
```

With default settings, the test run will take at most 100 seconds (60 seconds of ramping up traffic and then 40 seconds of waiting for responses), but in practice, the 60-second traffic phase will usually be cut short. To be precise, `resperf` can transition from the traffic-sending phase to the waiting-for-responses phase in three different ways:

- Running for the full allotted time and successfully reaching the maximum query rate (by default, 60 seconds and 100,000 qps, respectively). Since this is a very high query rate, this will rarely happen (with today's hardware); one of the other two conditions listed below will usually occur first.
- Exceeding 65,536 outstanding queries. This often happens as a result of (successfully) exceeding the capacity of the server being tested, causing the excess queries to be dropped. The limit of 65,536 queries comes from the number of possible values for the ID field in the DNS packet. `resperf` needs to allocate a unique ID for each outstanding query, and is therefore unable to send further queries if the set of possible IDs is exhausted.
- When `resperf` finds itself unable to send queries fast enough. `Resperf` will notice if it is falling behind in its scheduled query transmissions, and if this backlog reaches 1000 queries, it will print a message like "Fell behind by 1000 queries" (or whatever the actual number is at the time) and stop sending traffic.

Regardless of which of the above conditions caused the traffic-sending phase of the test to end, you should examine the resulting plots to make sure the server's response rate is flattening out toward the end of the test. If it is not, then you are not loading the server enough. If you are getting the "Fell behind" message, make sure that the machine running `resperf` is fast enough and has no other applications running.

You should also monitor the CPU usage of the server under test. It should reach close to 100% CPU at the point of maximum traffic; if it does not, you most likely have a bottleneck in some other part of your test setup, for example, your external Internet connection.

The report generated by `resperf-report` will be stored with a unique file name based on the current date and time, e.g., `20060812-1550.html`. The PNG images of the plots and

other auxiliary files will be stored in separate files beginning with the same date-time string. To view the report, simply open the .html file in a web browser.

If you need to copy the report to a separate machine for viewing, make sure to copy the .png files along with the .html file (or simply copy all the files, e.g., using `scp 20060812-1550.* host:directory/`).

4.24.2.4 Interpreting the report

The .html file produced by `resperf-report` consists of two sections. The first section, "Resperf output", contains output from the `resperf` program such as progress messages, a summary of the command line arguments, and summary statistics. The second section, "Plots", contains two plots generated by `gnuplot`: "Query/response/failure rate" and "Latency".

The "Query/response/failure rate" plot contains three graphs. The "Queries sent per second" graph shows the amount of traffic being sent to the server; this should be very close to a straight diagonal line, reflecting the linear ramp-up of traffic.

The "Total responses received per second" graph shows how many of the queries received a response from the server. All responses are counted, whether successful (NO-ERROR or NXDOMAIN) or not (e.g., SERVFAIL).

The "Failure responses received per second" graph shows how many of the queries received a failure response. A response is considered to be a failure if its RCODE is neither NOERROR nor NXDOMAIN.

By visually inspecting the graphs, you can get an idea of how the server behaves under increasing load. The "Total responses received per second" graph will initially closely follow the "Queries sent per second" graph (often rendering it invisible in the plot as the two graphs are plotted on top of one another), but when the load exceeds the server's capacity, the "Total responses received per second" graph may diverge from the "Queries sent per second" graph and flatten out, indicating that some of the queries are being dropped.

The "Failure responses received per second" graph will normally show a roughly linear ramp close to the bottom of the plot with some random fluctuation, since typical query traffic will contain some small percentage of failing queries randomly interspersed with the successful ones. As the total traffic increases, the number of failures will increase proportionally.

If the "Failure responses received per second" graph turns sharply upwards, this can be another indication that the load has exceeded the server's capacity. This will happen if the server reacts to overload by sending SERVFAIL responses rather than by dropping queries. Since Loop will respond with SERVFAIL when it exceeds its `recursive-clients` limit, a sudden increase in the number of failures could mean that the limit needs to be increased.

The "Latency" plot contains a single graph marked "Average latency". This shows how the latency varies during the course of the test. Typically, the latency graph will exhibit a downwards trend because the cache hit rate improves as ever more responses are cached during the test, and the latency for a cache hit is much smaller than for a

cache miss. The latency graph is provided as an aid in determining the point where the server gets overloaded, which can be seen as a sharp upwards turn in the graph. The latency graph is not intended for making absolute latency measurements or comparisons between servers; the latencies shown in the graph are not representative of production latencies due to the initially empty cache and the deliberate overloading of the server towards the end of the test.

Note that all measurements are displayed on the plot at the horizontal position corresponding to the point in time when the query was sent, not when the response (if any) was received. This makes it easy to compare the query and response rates; for example, if no queries are dropped, the query and response graphs will be identical. As another example, if the plot shows 10% failure responses at $t=5$ seconds, this means that 10% of the queries sent at $t=5$ seconds eventually failed, not that 10% of the responses received at $t=5$ seconds were failures.

4.24.2.5 Determining the server's maximum throughput

Often, the goal of running `resperf` is to determine the server's maximum throughput, in other words, the number of queries per second it is capable of handling. This is not always an easy task, because as a server is driven into overload, the service it provides may deteriorate gradually, and this deterioration can manifest itself either as queries being dropped, as an increase in the number of `SERVFAIL` responses, or an increase in latency. The maximum throughput may be defined as the highest level of traffic at which the server still provides an acceptable level of service, but that means you first need to decide what an acceptable level of service means in terms of packet drop percentage, `SERVFAIL` percentage, and latency.

The summary statistics in the "Resperf output" section of the report contains a "Maximum throughput" value which by default is determined from the maximum rate at which the server was able to return responses, without regard to the number of queries being dropped or failing at that point. This method of throughput measurement has the advantage of simplicity, but it may or may not be appropriate for your needs; the reported value should always be validated by a visual inspection of the graphs to ensure that service has not already deteriorated unacceptably before the maximum response rate is reached. It may also be helpful to look at the "Lost at that point" value in the summary statistics; this indicates the percentage of the queries that was being dropped at the point in the test when the maximum throughput was reached.

Alternatively, you can make `resperf` report the throughput at the point in the test where the percentage of queries dropped exceeds a given limit (or the maximum as above if the limit is never exceeded). This can be a more realistic indication of how much the server can be loaded while still providing an acceptable level of service. This is done using the `-L` command line option; for example, specifying `-L 10` makes `resperf` report the highest throughput reached before the server starts dropping more than 10% of the queries.

There is no corresponding way of automatically constraining results based on the number of failed queries, because unlike dropped queries, resolution failures will occur even when the server is not overloaded, and the number of such failures is heavily dependent on the query data and network conditions. Therefore, the plots

should be manually inspected to ensure that there is not an abnormal number of failures.

4.24.2.6 Generating constant traffic

In addition to ramping up traffic linearly, `resperf` also has the capability to send a constant stream of traffic. This can be useful when using `resperf` for tasks other than performance measurement; for example, it can be used to "soak test" a server by subjecting it to a sustained load for an extended period of time.

To generate a constant traffic load, use the `-c` command line option, together with the `-m` option which specifies the desired constant query rate. For example, to send 10000 queries per second for an hour, use `-m 10000 -c 3600`. This will include the usual 30-second gradual ramp-up of traffic at the beginning, which may be useful to avoid initially overwhelming a server that is starting with an empty cache. To start the onslaught of traffic instantly, use `-m 10000 -c 3600 -r 0`.

To be precise, `resperf` will do a linear ramp-up of traffic from 0 to `-m` queries per second over a period of `-r` seconds, followed by a plateau of steady traffic at `-m` queries per second lasting for `-c` seconds, followed by waiting for responses for an extra 40 seconds. Either the ramp-up or the plateau can be suppressed by supplying a duration of zero seconds with `-r 0` and `-c 0`, respectively. The latter is the default.

Sending traffic at high rates for hours on end will of course require very large amounts of input data. Also, a long-running test will generate a large amount of plot data, which is kept in memory for the duration of the test. To reduce the memory usage and the size of the plot file, consider increasing the interval between measurements from the default of 0.5 seconds using the `-i` option in long-running tests.

When using `resperf` for long-running tests, it is important that the traffic rate specified using the `-m` is one that both `resperf` itself and the server under test can sustain. Otherwise, the test is likely to be cut short as a result of either running out of query IDs (because of large numbers of dropped queries) or of `resperf` falling behind its transmission schedule.

4.24.3 The plot data file

The plot data file is written by the `resperf` program and contains the data to be plotted using `gnuplot`. When running `resperf` via the `resperf-report` script, there is no need for the user to deal with this file directly, but its format and contents are documented here for completeness and in case you wish to run `resperf` directly and use its output for purposes other than viewing it with `gnuplot`.

The first line of the file is a comment identifying the fields. It may be recognized as a comment by its leading hash sign (`#`).

Subsequent lines contain the actual plot data. For purposes of generating the plot data file, the test run is divided into time intervals of 0.5 seconds (or some other length of time specified with the `-i` command line option). Each line corresponds to one such interval, and contains the following values as floating-point numbers:

Time

The midpoint of this time interval, in seconds since the beginning of the run

Target queries per second

The number of queries per second scheduled to be sent in this time interval

Actual queries per second

The number of queries per second actually sent in this time interval

Responses per second

The number of responses received corresponding to queries sent in this time interval, divided by the length of the interval

Failures per second

The number of responses received corresponding to queries sent in this time interval and having an RCODE other than NOERROR or NXDOMAIN, divided by the length of the interval

Average latency

The average time between sending the query and receiving a response, for queries sent in this time interval

4.24.4 Options

Because the resperf-report script passes its command line options directly to the resperf programs, they both accept the same set of options, with one exception: resperf-report automatically adds an appropriate -P to the resperf command line, and therefore does not itself take a -P option.

-d `datafile`

Specifies the input data file. If not specified, resperf will read from standard input.

-s `server_addr`

Specifies the name or address of the server to which requests will be sent. The default is the loopback address, 127.0.0.1.

-p `port`

Sets the port on which the DNS packets are sent. If not specified, the standard DNS port (53) is used.

-a `local_addr`

Specifies the local address from which to send requests. The default is the wildcard address.

-x `local_port`

Specifies the local port from which to send requests. The default is the wildcard port (0).

If acting as multiple clients and the wildcard port is used, each client will use a different random port. If a port is specified, the clients will use a range of ports starting with the specified one.

-t `timeout`

Specifies the request timeout value, in seconds. `resperf` will no longer wait for a response to a particular request after this many seconds have elapsed. The default is 45 seconds.

`resperf` times out unanswered requests in order to reclaim query IDs so that the query ID space will not be exhausted in a long-running test, such as when "soak testing" a server for an day with `-m 10000 -c 86400`. The timeouts and the ability to tune them are of little use in the more typical use case of a performance test lasting only a minute or two.

The default timeout of 45 seconds was chosen to be longer than the query timeout of current caching servers. Note that this is longer than the corresponding default in `dnssperf`, because caching servers can take many orders of magnitude longer to answer a query than authoritative servers do.

If a short timeout is used, there is a possibility that `resperf` will receive a response after the corresponding request has timed out; in this case, a message like `Warning: Received a response with an unexpected id: 141` will be printed.

-b `bufsize`

Sets the size of the socket's send and receive buffers, in kilobytes. If not specified, the operating system's default is used.

-f `family`

Specifies the address family used for sending DNS packets. The possible values are "inet", "inet6", or "any". If "any" (the default value) is specified, `resperf` will use whichever address family is appropriate for the server it is sending packets to.

-e

Enables EDNS0 [RFC2671], by adding an OPT record to all packets sent.

-D

Sets the DO (DNSSEC OK) bit [RFC3225] in all packets sent. This also enables EDNS0, which is required for DNSSEC.

-y `[alg:]name:secret`

Add a TSIG record [RFC2845] to all packets sent, using the specified TSIG key algorithm, name and secret, where the algorithm defaults to `hmac-md5` and the secret is expressed as a base-64 encoded string.

-h

Print program usage information and exit.

-i `interval`

Specifies the time interval between data points in the plot file. The default is 0.5 seconds.

-m `max_qps`

Specifies the target maximum query rate (in queries per second). This should be higher than the expected maximum throughput of the server being tested. Traffic will be ramped up at a linearly increasing rate until this value is reached, or until one of the other conditions described in the section "Running the test" occurs. The default is 100000 queries per second.

-P `plot_data_file`

Specifies the name of the plot data file. The default is `resperf.gnuplot`.

-r `rampup_time`

Specifies the length of time over which traffic will be ramped up. The default is 60 seconds.

-c `constant_traffic_time`

Specifies the length of time for which traffic will be sent at a constant rate following the initial ramp-up. The default is 0 seconds, meaning no sending of traffic at a constant rate will be done.

-L `max_loss`

Specifies the maximum acceptable query loss percentage for purposes of determining the maximum throughput value. The default is 100%, meaning that `resperf` will measure the maximum throughput without regard to query loss.

-C `clients`

Act as multiple clients. Requests are sent from multiple sockets. The default is to act as 1 client.

-q `max_outstanding`

Sets the maximum number of outstanding requests. `resperf` will stop ramping up traffic when this many queries are outstanding. The default is 64k, and the limit is 64k per client.

-v

Print program version and exit.

4.24.5 See also

`dnssperf(1)`

4.24.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (C) 2000, 2001, 2004-2015 Nominum, Inc.

CONFIGURATION

The following sections document configuration files of some programs that are part of the Loop software distribution.

5.1 `named.conf` --- `named` program's configuration

5.1.1 Description

`named.conf` is the configuration file for `named(8)`, the DNS nameserver daemon.

5.1.2 Configuration grammar

The configuration file consists of configuration statements and comments. Statements end with a semicolon. Statements and comments are the only elements that can appear without enclosing braces. Many statements contain a block of sub-statements, which are also terminated with a semicolon. Clauses in the statements are also semi-colon terminated. See the [Comments syntax](#) section for a description of comments, and the Loop User Manual for some examples.

`named.conf` supports the following statements:

acl

Defines a named IP address matching list, for access control and other uses.

controls

Declares remote control channels to be used by **rndc** to communicate with a **named** process.

include

Includes a config file.

key

Specifies key information for use in authentication and authorization with TSIG, or the **rndc** control channel.

logging

Specifies what the server logs, and where.

managed-keys

Specifies DNSSEC trust anchors to be kept up-to-date using RFC 5011 trust anchor maintenance.

masters

Defines a named masters list for inclusion in stub and slave zones' `masters` and `also-notify` lists.

options

Specifies global nameserver config options, which also sets the default values for some config options at the `view` and `zone` statements.

server

Specifies some config options on a per-server basis.

trusted-keys

Specifies static DNSSEC trust anchors.

view

Defines a view, and specifies some config options at that view level.

zone

Defines a zone, and specifies some config options at that zone level.

Note

The `logging` and `options` statements may occur only once in the configuration file.

5.1.2.1 Data types

Following is a list of data types used throughout the Loop configuration file:

`<address_match_element>`

`<address_match_element>` lists are primarily used to control access for various nameserver operations. They are also used in the `listen-on` statement. The elements which constitute an `<address_match_element>` list can be any of the following:

- An IP address (IPv4 or IPv6)
- An IP prefix (in ``/'` notation)
- A key ID, as defined by the `key` statement
- The name of an `<address_match_element>` list defined with the `acl` statement
- A nested `<address_match_element>` list enclosed in braces

`<address_match_element>`s can be negated with `!` (a leading exclamation mark). Some pre-defined named `<address_match_element>` lists can be found in the description of the `acl` statement.

The addition of the `key` clause made the name of this type something of a misnomer, as security keys can be used to validate access without regard to an IP address. Nonetheless, the type is still called `<address_match_element>`.

When a given IP address or prefix is compared to an `<address_match_element>` list, the matching takes place in approximately $O(1)$ time. However, key matching requires that the list of keys be traversed until a matching key is found, and therefore may be somewhat slower as it is an $O(N)$ operation.

The interpretation of a match depends on whether the list is being used for access-control, or defining `listen-on` ports, and whether the `<address_match_element>` was negated.

When used as an access control list, a non-negated match allows access and a negated match denies access. If there is no match, access is denied. The clauses `allow-notify`, `allow-recursion`, `allow-recursion-on`, `allow-query`, `allow-query-on`, `allow-query-cache`, `allow-query-cache-on`, `allow-transfer`, `allow-update`, `allow-update-forwarding`, and `block` all use `<address_match_element>` lists. Similarly, the `listen-on` option will cause the server to refuse queries on any of the machine's addresses which do not match the list.

The order of insertion is significant. If more than one `<address_match_element>` in an ACL is found to match a given IP address or prefix, preference will be given to the one that came *first* in the ACL definition. Because of this first-match behavior, an element that defines a subset of another element in the list should come before the broader element, regardless of whether either is negated. For example, in `1.2.3/24; ! 1.2.3.13;` the `1.2.3.13` element is completely useless because the algorithm will match any lookup for `1.2.3.13` to the `1.2.3/24` element. Using `! 1.2.3.13; 1.2.3/24` fixes that problem by having `1.2.3.13` blocked by the negation, but all other `1.2.3.*` hosts fall through.

<boolean>

A boolean type. Takes the values `yes` or `no`. The values `true` and `false` are also accepted, as are the numbers `1` and `0`.

`fixedpoint`

`integer`

`log_severity`

`netprefix`

`portrange`

`quoted_string`

`rrtypelist`

server_key
size_no_default
sizeval
string
ttlval

Error

TODO: Complete the list of data types.

5.1.2.2 acl statement

```
acl <name:string> { <value:address_match_element>; ... }; //   
↪may occur multiple times
```

The `acl` statement defines a named access control list (ACL). It assigns the given list of address match elements to the symbolic name specified by *name*. The `acl` statement gets its name from a primary use of address match elements which is in access control lists (ACLs).

The following ACL names are built into **named**:

- `any` matches all hosts.
- `none` matches no hosts.
- `localhost` matches the IPv4 and IPv6 addresses of all network interfaces on the system. When addresses are added or removed, the `localhost` ACL element is updated to reflect the changes.
- `localnets` matches any host on an IPv4 or IPv6 network for which the system has an interface. When addresses are added or removed, the `localnets` ACL element is updated to reflect the changes. Some systems do not provide a way to determine the prefix lengths of local IPv6 addresses. In such a case, `localnets` only matches the local IPv6 addresses, just like `localhost`.

5.1.2.3 controls statement

```
controls {  
    inet ( <ipv4_address> | <ipv6_address> | * ) [ port (   
↪<integer> | * ) ] allow { <allow:address_match_element>; ...  
↪ } [ keys { <keys:string>; ... } ]; // may occur multiple_  
↪times  
}; // may occur multiple times
```

The `controls` statement declares control channels to be used by system administrators to control the operation of **named**. These control channels are used by the **rndc** program to send commands to and retrieve results from **named**.

An `inet` control channel is a TCP socket listening at the specified port number and IP address (which can be an IPv4 or IPv6 address). An IP address specified as `*` (asterisk) is interpreted as the IPv4 wildcard address; connections will be accepted on any of the system's IPv4 addresses. To listen on the IPv6 wildcard address, use an IP address specified as `::`. If you will only use **rndc** on the localhost, using the loopback address (127.0.0.1 or `::1`) is recommended for maximum security.

If no port is specified, port number 953 is used by default. `*` cannot be used for the port number.

Error

TODO: The grammar allows `*` for the port, so this should be fixed to default to 953 when `*` is specified for the port.

The ability to issue commands over the control channel is restricted by the `allow` and `keys` clauses. Connections to the control channel are permitted based on the `address_match_element` list. This is for simple IP address based filtering only; any key elements in the `address_match_element` list are ignored.

The primary authorization mechanism of the control channel is the `keys` list, which contains a list of `key_ids`. Each `key_id` in the `keys` list is authorized to execute commands over the control channel. See `rndc.conf(5)` for information about configuring keys for **rndc**.

If no `controls` statement is present, **named** will set up a default control channel listening on the loopback address 127.0.0.1 and its IPv6 counterpart `::1`. In this case, and also when the `controls` statement is present but does not have a `keys` clause, **named** will attempt to load the command channel key from the file `/etc/loop/rndc.key`. To create this file, run:

```
$ rndc-confgen -a
```

The `rndc.key` feature does not have a high degree of configurability. You cannot easily change the key name or the size of the secret, so you should make an `rndc.conf(5)` configuration file with your own key if you wish to change those settings. The `rndc.key` file also has its permissions set such that only the owner of the file (the user that **named** is running as) can access it. If you desire greater flexibility in allowing other users to run **rndc** commands, then you need to create an `rndc.conf(5)` file and make it group readable by a group that contains the users who should have access.

To disable the command channel, use an empty `controls` statement:

```
controls {};
```

5.1.2.4 include statement

```
include <quoted_string>;
```

The `include` statement inserts the specified file at the point where the `include` statement is encountered. Glob patterns such as `*` are supported in the filename, and all matching files are included.

The `include` statement facilitates the administration of configuration files by permitting the reading or writing of some files but not others. For example, the statement could include files with keys that are readable only by the user the nameserver is running as.

5.1.2.5 key statement

```
key <string> {  
    algorithm <string>;  
    secret <string>;  
}; // may occur multiple times
```

The `key` statement defines a shared secret key for use with TSIG or the control channel (see [controls statement](#)). The `key` statement can occur at the top level of the configuration file, or within a `view` statement. Keys defined in top-level `key` statements can be used in all views. Keys intended for use in a `controls` statement must be defined at the top-level.

After the `key` keyword, the `key` statement contains a string which is the key ID (also known as a key name) which is a DNS name that uniquely identifies the key. It can be used in a `server` statement to cause requests sent to that server to be TSIG-signed with this key, or in address match lists to verify that incoming requests have been signed with a key matching this name, algorithm, and secret.

The `algorithm` clause contains a string specifying the HMAC algorithm. **named** supports `hmac-md5`, `hmac-sha1`, `hmac-sha224`, `hmac-sha256`, `hmac-sha384` and `hmac-sha512` algorithms. Truncated hashes are supported by appending the minimum number of required bits preceded by a dash, e.g. `hmac-sha1-80`.

Error

TODO: Remove support for truncated hashes.

The `secret` clause specifies the secret to be used by the HMAC algorithm in Base64 encoding ([RFC 4648](#)).

The **ddns-confgen** program can be used with the `-q` argument to generate just a `key` statement:

```
$ ddns-confgen -q  
key "ddns-key" {
```

(continues on next page)

(continued from previous page)

```
algorithm hmac-sha256;
secret "gxSW0mdkZ9o90OuxruPiqbxiaZjKyZ9+cst/cRu8Np0=";
};
$
```

5.1.2.6 logging statement

```
logging {
    channel <string> {
        file <quoted_string> [ versions ( "unlimited"
↪ | <integer> ) ] [ size <size> ];
        syslog [ <syslog_facility> ];
        null;
        stderr;
        severity <log_severity>;
        print-time <boolean>;
        print-severity <boolean>;
        print-category <boolean>;
    }; // may occur multiple times
    category <name:string> { <destinations:string>; ... };
↪ // may occur multiple times
};
```

The logging statement configures a wide variety of logging options for **named**.. Its **channel** phrase associates output methods, format options and severity levels with a name that can then be used with the **category** phrase to select how various classes of messages are logged.

Only one logging statement is to be used to define as many channels and categories as are wanted. If there is no logging statement, the logging configuration will be:

```
logging {
    category default { default_syslog; default_debug; };
    category unmatched { null; };
};
```

Note

The logging statement may occur only once in the configuration file.

In Loop, the logging configuration is only established when the entire configuration file has been parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels, or to standard error if the **named -g** argument was specified.

5.1.2.6.1 `channel` phrase

All log output goes to one or more *channels*; you can make as many of them as you want.

Every channel definition must include a destination clause that says whether messages selected for the channel go to a file, or to a particular syslog facility, or to the standard error stream, or are discarded. It can optionally also limit the message severity level that will be accepted by the channel (the default is `info`), and whether to include a **named**-generated time stamp, the category name, and/or severity level. The default is not to include any.

file <quoted_string> [`versions` ("unlimited" | <integer>)] [`size` <size>

The `file` destination clause directs the channel to a disk file. It can include limitations both on how large the file is allowed to become, and how many versions of the file will be saved each time the file is opened.

If you use the `versions` log file option, then **named** will retain that many backup versions of the file by renaming them when opening. For example, if you choose to keep three old versions of the file `lamers.log`, then just before it is opened `lamers.log.1` is renamed to `lamers.log.2`, `lamers.log.0` is renamed to `lamers.log.1`, and `lamers.log` is renamed to `lamers.log.0`. You can use `versions unlimited` to not limit the number of versions. If a `size` option is associated with the log file, then renaming is only done when the file being opened exceeds the indicated size. No backup versions are kept by default; any existing log file is simply appended to.

The `size` option for files is used to limit log growth. If the file ever exceeds the size, then **named** will stop writing to the file unless it has a `versions` option associated with it. If backup versions are kept, the files are rolled as described above and a new one begun. If there is no `versions` option, no more data will be written to the log until some out-of-band mechanism removes or truncates the log to less than the maximum size. The default behavior is not to limit the size of the file.

Example usage of the `size` and `versions` options:

```
channel an_example_channel {
    file "example.log" versions 3 size 20m;
    print-time yes;
    print-category yes;
};
```

syslog [<syslog_facility>]

The `syslog` destination clause directs the channel to the system log (see *syslog(3)*). Its argument is a syslog facility as described in the *openlog(3)* manpage. Known facilities are `kern`, `user`, `mail`, `daemon`, `auth`, `syslog`, `lpr`, `news`, `uucp`, `cron`, `authpriv`, `ftp`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6` and `local7`, however not all facilities are supported on all operating systems. How the syslog daemon will handle messages sent to this facility would be described in its documentation.

null

The `null` destination clause causes all messages sent to the channel to be discarded; in that case, other options for the channel are meaningless.

stderr

The `stderr` destination clause directs the channel to the server's standard error stream. This is intended for use when the server is running as a foreground process, for example, when the administrator is debugging a configuration.

The server can supply extensive debugging information when it is in debugging mode. If the server's global debug level is greater than zero, then debugging mode will be active. The global debug level is set either by starting **named** with the `-d` argument followed by a positive integer, or by running the `rndc trace` command. The global debug level can be set to zero, and debugging mode turned off, by running `rndc notrace`. All debugging messages in the server have a debug level, and higher debug levels give more detailed output. Channels that specify a specific debug severity, for example:

```
channel specific_debug_level {
    file "foo";
    severity debug 3;
};
```

will get debugging output of level 3 or less any time the server is in debugging mode, regardless of the global debugging level. Channels with `dynamic severity` use the server's global debug level to determine what messages to print.

severity <log_severity>

The `severity` clause works like *syslog(3)*'s "priorities", except that they can also be used if you are writing straight to a file rather than using `syslog`. Messages which are not at least of the severity level given will not be selected for the channel; messages of higher severity levels will be accepted.

If you are using `syslog`, then the `syslog` daemon configuration's priorities will also determine what eventually passes through. For example, defining a channel facility and severity as `daemon` and `debug` but only logging `daemon.warning` via the `syslog` daemon configuration's will cause messages of severity `info` and `notice` to be dropped. If the situation were reversed, with **named** writing messages of only `warning` or higher, then the `syslog` daemon would allow all messages it received from the channel.

print-time <boolean>

If `print-time` is enabled, then the date and time will be logged. `print-time` may be specified for a `syslog` channel, but is usually pointless since `syslog` also logs the date and time.

print-severity <boolean>

If `print-severity` is enabled, then the severity level of the message will be logged.

print-category <boolean>

If `print-category` is enabled, then the category of the message will be logged as well.

The `print-time`, `print-severity`, and `print-category` options may be used in any combination, and will always be printed in the following order: time, category, severity. Here is an example where all the three options are enabled:

```
28-Feb-2000 15:05:32.863 general: notice: running
```

There are four predefined channels that are used for **named**'s default logging as follows. (See *category phrase* for how they are used.)

```
channel default_syslog {
    // send to syslog's daemon facility
    syslog daemon;
    // only send priority info and higher
    severity info;
};

channel default_debug {
    // write to named.run in the working directory
    // Note: stderr is used instead of "named.run" if
    // the server is started with the '-f' option.
    file "named.run";
    // log at the server's current debug level
    severity dynamic;
};

channel default_stderr {
    // writes to stderr
    stderr;
    // only send priority info and higher
    severity info;
};

channel null {
    // toss anything sent to this channel
    null;
};
```

The `default_debug` channel has the special property that it only produces output when the server's debug level is nonzero. It normally writes to a file called `named.run` in the server's working directory.

For security reasons, when the **named** `-u` argument is used, the `named.run` file is created only after **named** has changed to the new UID, and any debug output generated while **named** is starting up and still running as root is discarded. If you need to capture this output, you must run **named** with the `-g` argument and redirect standard

error to a file.

Once a channel is defined, it cannot be redefined. Thus you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined.

5.1.2.6.2 category phrase

There are many categories, so you can send the logs you want to see wherever you want, without seeing logs you don't want. If you don't specify a list of channels for a category, then log messages in that category will be sent to the `default` category instead. If you don't specify a default category, the following "default default" is used:

```
category default { default_syslog; default_debug; };
```

As an example, let's say you want to log security events to a file, but you also want keep the default logging behavior. You'd specify the following:

```
channel my_security_channel {
    file "my_security_file";
    severity info;
};

category security {
    my_security_channel;
    default_syslog;
    default_debug;
};
```

To discard all messages in a category, specify the `null` channel:

```
category xfer-out { null; };
category notify { null; };
```

The following is a list of available categories and brief descriptions of the types of log information they contain. More categories may be added in future Loop releases.

client

Processing of client requests.

cname

Logs nameservers that are skipped due to them being a CNAME rather than A / AAAA records.

config

Configuration file parsing and processing.

database

Messages relating to the databases used internally by the name server to store zone and cache data.

default

The default category defines the logging options for those categories where no specific configuration has been defined.

delegation-only

Delegation-only. Logs queries that have been forced to NXDOMAIN as the result of a delegation-only zone or a `delegation-only` in a forward, hint or stub zone declaration.

dispatch

Dispatching of incoming packets to the server modules where they are to be processed.

dnssec

DNSSEC and TSIG protocol processing.

edns-disabled

Log queries that have been forced to use plain DNS due to timeouts. This is often due to the remote servers not being [RFC 1034](#) compliant (not always returning FORMERR or similar to EDNS queries and other extensions to the DNS when they are not understood). In other words, this is targeted at servers that fail to respond to DNS queries that they don't understand.

Note

The log message can also be due to packet loss. Before reporting servers for RFC 1034 non-compliance they should be re-tested to determine the nature of the non-compliance. This testing should prevent or reduce the number of false-positive reports.

Note

Eventually **named** will have to stop treating such timeouts as due to RFC 1034 non compliance and start treating it as plain packet loss. Falsely classifying packet loss as due to RFC 1034 non-compliance impacts on DNSSEC validation which requires EDNS for the DNSSEC records to be returned.

general

The catch-all category. Many messages still aren't classified into categories, and they will end up in the `general` category.

lame-servers

Lame servers. These are misconfigurations in remote servers, discovered by Loop when trying to query those servers during resolution.

network

Network operations.

notify

DNS NOTIFY related.

queries

Specify where queries should be logged to.

At startup, specifying the category `queries` will also enable query logging unless the `querylog` option has been configured.

The query log entry reports the client's IP address, port number, the query name, class, and type. Next it reports whether the RD (Recursion Desired) flag was set (+ if set, - if not set), if the query was signed (S), if EDNS was used (E), if TCP was used (T), if DO (DNSSEC Ok) flag was set (D), or if CD (Checking Disabled) flag was set (C). After this, the destination address the query was sent to is reported.

For example,

```
client 127.0.0.1#62536 (www.example.com): query: www.
↳example.com IN AAAA +SE
client ::1#62537 (www.example.net): query: www.example.
↳net IN AAAA -SE
```

The first part of this log message, showing the client address/port number and query name, is repeated in all subsequent log messages related to the same query.

query-errors

Information about queries that resulted in some failure. The `query-errors` category is specifically intended for debugging purposes: To identify why and how specific queries result in responses which indicate an error. Messages of this category are therefore only logged with debug levels.

At the debug levels of 1 or higher, each response with the rcode of SERVFAIL is logged as follows:

```
client 127.0.0.1#61502: query failed (SERVFAIL) for www.
↳example.com/IN/AAAA at query.c:3880
```

This means an error resulting in SERVFAIL was detected at line 3880 of source code file `query.c`. Log messages of this level will particularly help identify the cause of SERVFAIL for an authoritative server.

At the debug levels of 2 or higher, detailed context information of recursive resolutions that resulted in SERVFAIL is logged. The log message will look like as follows:

```
fetch completed at resolver.c:2970 for www.example.com/A_
↳in 30.000183: timed out/success [domain:example.com,
↳referral:2,restart:7,qrysent:8,timeout:5,lame:0,
↳neterr:0,badresp:1,adberr:0,findfail:0,valfail:0]
```

The first part before the colon shows that a recursive resolution for AAAA records of `www.example.com` completed in 30.000183 seconds and the final result that led to the SERVFAIL was determined at line 2970 of source code file

`resolver.c.`

The next part of the log message shows the detected final result and the latest result of DNSSEC validation. The latter is always success when no validation attempt is made. In this example, this query resulted in SERVFAIL probably because all name servers are down or unreachable, leading to a timeout in 30 seconds. DNSSEC validation was probably not attempted.

The last part of the log message enclosed in square brackets shows statistics information collected for this particular resolution attempt. The meaning of these fields is summarized below:

`domain`

The `domain` field shows the deepest zone that the resolver reached; it is the zone where the error was finally detected. The meaning of the other fields is summarized in the following table.

`referral`

The number of referrals the resolver received throughout the resolution process. In the above example this is 2, which are most likely *com.* and *example.com.*

`restart`

The number of cycles that the resolver tried remote nameservers at the `domain` zone. In each cycle the resolver sends one query (possibly resending it, depending on the response) to each known nameserver of the `domain` zone.

`qrysent`

The number of queries the resolver sent at the `domain` zone.

`timeout`

The number of timeouts since the resolver received the last response.

`lame`

The number of lame servers the resolver detected at the `domain` zone. A server is detected to be lame by an invalid response.

`neterr`

The number of erroneous results that the resolver encountered in sending queries at the `domain` zone. One common case is the remote server is unreachable and the resolver receives an ICMP unreachable error message.

`badresp`

The number of unexpected responses (other than `lame`) to queries sent by the resolver at the `domain` zone.

`adberr`

Failures in finding remote nameserver addresses of the `domain` zone in the ADB. One common case of this is that the remote nameserver's hostname does not have any address records.

`findfail`

Failures of resolving remote server addresses. This is a total number of failures throughout the resolution process.

`valfail`

Failures of DNSSEC validation. Validation failures are counted throughout the resolution process (not limited to the `domain` zone), but should only happen in `domain`.

At the debug levels of 3 or higher, the same messages as those at the debug 1 level are logged for other errors than `SERVFAIL`. Note that negative responses such as `NXDOMAIN` are not regarded as errors here.

At the debug levels of 4 or higher, the same messages as those at the debug 2 level are logged for other errors than `SERVFAIL`. Unlike the above case of level 3, messages are logged for negative responses. This is because any unexpected results can be difficult to debug in the recursion case.

rate-limit

The start, periodic, and final notices of the rate-limiting of a stream of responses are logged at `info` severity in this category. These messages include a hash value of the domain name of the response and the name itself, except when there is insufficient memory to record the name for the final notice. The final notice is normally delayed until about one minute after rate limit stops. A lack of memory can hurry the final notice, in which case it starts with `*` (asterisk). Various internal events are logged at debug level 1 and higher.

Rate limiting of individual requests is logged in the `query-errors` category.

resolver

DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.

rpz

Information about errors in response policy zone files, rewritten responses, and at the highest debug levels, mere rewriting attempts.

security

Approval and denial of requests.

spill

Logs queries that have been terminated, either by dropping or responding with `SERVFAIL`, as a result of a `fetchlimit` quota being exceeded.

unmatched

Messages that **named** was unable to determine the class of or for which there was no matching `view`. A one line summary is also logged to the `client` category.

This category is best sent to a file or stderr, by default it is sent to the `null` channel.

update

DNS UPDATE related.

update-security

Approval and denial of update requests.

xfer-in

Zone transfers the server is receiving.

xfer-out

Zone transfers the server is sending.

5.1.2.7 managed-keys statement

```
managed-keys { <name:string> <init:string> <flags:integer>  
  ↪ <protocol:integer> <algorithm:integer> <key:quoted_string>;  
  ↪ ... }; // may occur multiple times
```

The `managed-keys` statement (like the *trusted-keys statement*), defines DNSSEC trust anchors. The difference is that `managed-keys` specifies dynamic trust anchors that can be kept up-to-date automatically, without intervention from the resolver operator (see [RFC 5011](#)).

Suppose, for example, that a zone's key-signing key (KSK) was compromised, and the zone owner had to revoke and replace the key. A resolver which had the old key configured as a trust anchor in a `trusted-keys` statement would be unable to validate this zone any longer; it would reply with a SERVFAIL response code. This would continue until the resolver operator had updated the `trusted-keys` statement with the new key.

If, however, the zone were listed in a `managed-keys` statement instead, then the zone owner could add a "stand-by" key to the zone in advance. **named** would retrieve and store the stand-by key, and when the original key was revoked, **named** would be able to transition smoothly to the new trust anchor. It would also recognize that the old key had been revoked, and cease using that key as a trust anchor to validate answers, minimizing the damage that the compromised key could do.

A `managed-keys` statement contains a list of the keys to be managed, along with information about how the keys are to be initialized for the first time using the `init` field. The only initialization method currently supported is `initial-key`. This means the `managed-keys` statement must contain a copy of the initializing key. (Future releases may allow keys to be initialized by other methods, eliminating this requirement.)

Consequently, a `managed-keys` statement appears similar to a `trusted-keys` statement, differing in the presence of the second field containing the keyword `initial-key`. The difference is that, whereas the keys listed in a `trusted-keys` continue to be trusted until they are removed from the config file, an initializing key

listed in a `managed-keys` statement is only trusted *once*: for as long as it takes to load the managed key database and start the RFC 5011 key maintenance process.

The first time **named** runs with a managed key configured in the config file, it fetches the DNSKEY RRset directly from the zone's apex, and validates it using the key specified in the `managed-keys` statement. If the DNSKEY RRset is validly signed, then it is used as the basis for a new managed keys database.

From that point on, whenever **named** runs, it sees the `managed-keys` statement and checks to make sure RFC 5011 key maintenance has already been initialized for the specified domain, and if so, it simply moves on. The key specified in the `managed-keys` statement in the config file is not used to validate answers; it has been superseded by the key or keys stored in the managed keys database.

The next time **named** runs after a name has been *removed* from the `managed-keys` statement, the corresponding zone will be removed from the managed keys database, and RFC 5011 key maintenance will no longer be used for that domain.

In the current implementation, the managed keys database is stored in a master-format zone file using experimental RR types.

On servers which do not use views, this file is named `managed-keys.loop`. When views are in use, there will be a separate managed keys database for each view; the filename will be a hash of the view name followed by the suffix `.mkeys`.

When the key database is changed, the zone is updated. As with any other dynamic zone, changes will be written into a journal file, e.g., `managed-keys.loop.jnl`. Changes are committed to the master file as soon as possible afterward; this will usually occur within 30 seconds. So, whenever **named** is using automatic key maintenance, the zone file and journal file can be expected to exist in the working directory.

Note

Unless the `managed-keys-directory` option is configured, the working directory (see the `directory` option) should be writable by **named** so that it can write to the managed-keys database.

If the `dnssec-validation` option is set to `auto`, **named** will automatically initialize a managed key for the root zone. The key that is used to initialize the key maintenance process is built-in and can be overridden with the `dnssec-keys-file` option.

5.1.2.8 masters statement

```
masters <name:string> [ port <port:integer> ] [ dscp
↳<dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↳<integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↳<key:string> ]; ... }; // may occur multiple times
```

The `masters` statement can be used to specify a common set of masters (a masters list) that can be used by multiple stub and slave zones in their `masters` or `also-notify`

lists. It is a convenience feature so that the same masters need not be specified multiple times.

Error

TODO: Expand this description, and add examples.

5.1.2.9 options statement

```
options {
    automatic-interface-scan <boolean>;
    avoid-v4-udp-ports { <portrange>; ... };
    avoid-v6-udp-ports { <portrange>; ... };
    dnssec-keys-file <quoted_string>;
    block { <address_match_element>; ... };
    coresize ( default | unlimited | <sizeval> );
    datasize ( default | unlimited | <sizeval> );
    directory <quoted_string>;
    dscp <integer>;
    dump-file <quoted_string>;
    files ( default | unlimited | <sizeval> );
    flush-zones-on-shutdown <boolean>;
    heartbeat-interval <integer>;
    interface-interval <integer>;
    listen-on [ port <port:integer> ] [ dscp
↵<dscp:integer> ] { <acl:address_match_element>; ... }; //↵
↵may occur multiple times
        listen-on-v6 [ port <port:integer> ] [ dscp
↵<dscp:integer> ] { <acl:address_match_element>; ... }; //↵
↵may occur multiple times
        managed-keys-directory <quoted_string>;
        match-mapped-addresses <boolean>;
        max-rsa-exponent-size <integer>;
        pid-file ( <quoted_string> | none );
        port <integer>;
        querylog <boolean>;
        recursing-file <quoted_string>;
        recursive-clients <integer>;
        reserved-sockets <integer>;
        secroots-file <quoted_string>;
        serial-query-rate <integer>;
        server-id ( <quoted_string> | none | hostname );
        session-keyalg <string>;
        session-keyfile ( <quoted_string> | none );
        session-keyname <string>;
        cookie-algorithm ( hmac-sha256 | siphash );
```

(continues on next page)

(continued from previous page)

```

cookie-secret <string>;
stacksize ( default | unlimited | <sizeval> );
statistics-file <quoted_string>;
tcp-clients <integer>;
tcp-listen-queue <integer>;
tkey-dhkey <name:quoted_string> <keyid:integer>;
tkey-domain <quoted_string>;
tkey-gssapi-credential <quoted_string>;
tkey-gssapi-keytab <quoted_string>;
transfers-in <integer>;
transfers-out <integer>;
transfers-per-ns <integer>;
use-v4-udp-ports { <portrange>; ... };
use-v6-udp-ports { <portrange>; ... };
version ( <quoted_string> | none );
allow-new-zones <boolean>;
allow-query-cache { <address_match_element>; ... };
allow-query-cache-on { <address_match_element>; ... };
allow-recursion { <address_match_element>; ... };
allow-recursion-on { <address_match_element>; ... };
attach-cache <string>;
auth-nxdomain <boolean>; // default changed
cache-file <quoted_string>; // test only
check-names ( master | slave | response ) ( fail |
warn | ignore ); // may occur multiple times
clients-per-query <integer>;
deny-answer-addresses { <acl:address_match_element>; .
... } [ except-from { <except-from:quoted_string>; ... } ];
deny-answer-aliases { <name:quoted_string>; ... } [
except-from { <except-from:quoted_string>; ... } ];
disable-algorithms <domain:string> {
<algorithms:string>; ... }; // may occur multiple times
disable-ds-digests <domain:string> { <digests:string>;
... }; // may occur multiple times
disable-empty-zone <string>; // may occur multiple
times
dns64 <netprefix> {
    break-dnssec <boolean>;
    clients { <address_match_element>; ... };
    exclude { <address_match_element>; ... };
    mapped { <address_match_element>; ... };
    recursive-only <boolean>;
    suffix <ipv6_address>;
}; // may occur multiple times
dns64-contact <string>;
dns64-server <string>;

```

(continues on next page)

(continued from previous page)

```

    dnssec-enable <boolean>;
    dnssec-must-be-secure <domain:string> <value:boolean>;
↪ // may occur multiple times
    dnssec-validation ( yes | no | auto );
    dual-stack-servers [ port <port:integer> ] { (
↪<quoted_string> [ port <integer> ] [ dscp <integer> ] |
↪<ipv4_address> [ port <integer> ] [ dscp <integer> ] |
↪<ipv6_address> [ port <integer> ] [ dscp <integer> ] ); ...↪
↪};

    edns-udp-size <integer>;
    empty-contact <string>;
    empty-server <string>;
    empty-zones-enable <boolean>;
    fetch-quota-params <frequency:integer>
↪<low:fixedpoint> <high:fixedpoint> <discount:fixedpoint>;
    fetches-per-server <fetches:integer> [ ( drop | fail↪
↪) ];
    fetches-per-zone <fetches:integer> [ ( drop | fail )↪
↪];

    ixfr-from-differences ( master | slave | <boolean> );
    max-cache-size <size_no_default>;
    max-cache-ttl <integer>;
    max-clients-per-query <integer>;
    max-ncache-ttl <integer>;
    max-recursion-depth <integer>;
    max-recursion-queries <integer>;
    max-udp-size <integer>;
    minimal-responses <boolean>;
    no-case-compress { <address_match_element>; ... };
    nocookie-udp-size <integer>;
    preferred-glue <string>;
    prefetch <trigger:integer> [ <integer> ];
    provide-ixfr <boolean>;
    query-source ( ( [ address ] ( <ipv4_address> | * ) [↪
↪port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_↪
↪address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>↪
↪ ];
    query-source-v6 ( ( [ address ] ( <ipv6_address> | *↪
↪) [ port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_↪
↪address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>↪
↪ ];

    rate-limit {
        all-per-second <integer>;
        errors-per-second <integer>;
        exempt-clients { <address_match_element>; ...↪
↪};

```

(continues on next page)

(continued from previous page)

```

        ipv4-prefix-length <integer>;
        ipv6-prefix-length <integer>;
        log-only <boolean>;
        max-table-size <integer>;
        min-table-size <integer>;
        nodata-per-second <integer>;
        nxdomains-per-second <integer>;
        qps-scale <integer>;
        referrals-per-second <integer>;
        responses-per-second <integer>;
        slip <integer>;
        window <integer>;

    };
    recursion <boolean>;
    request-nsid <boolean>;
    request-cookie <boolean>;
    resolver-query-timeout <integer>;
    response-policy { zone <quoted_string> [ policy (
↪cname | disabled | drop | given | no-op | nodata | nxdomain_
↪| passthru | tcp-only <cname:quoted_string> ) ] [ recursive-
↪only <boolean> ] [ max-policy-ttl <integer> ]; ... } [
↪recursive-only <boolean> ] [ break-dnssec <boolean> ] [ max-
↪policy-ttl <integer> ] [ min-ns-dots <integer> ] [ qname-
↪wait-recurse <boolean> ];
        root-delegation-only [ exclude { <quoted_string>; ...
↪} ];
    rrset-order ( none | random );
    zero-no-soa-ttl-cache <boolean>;
    allow-notify { <address_match_element>; ... };
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    allow-transfer { <address_match_element>; ... };
    allow-update { <address_match_element>; ... };
    allow-update-forwarding { <address_match_element>; ...
↪ };
        also-notify [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↪<integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↪<key:string> ]; ... };
        alt-transfer-source ( <ipv4_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
        alt-transfer-source-v6 ( <ipv6_address> | * ) [ port
↪( <integer> | * ) ] [ dscp <integer> ];
        auto-dnssec ( allow | maintain | off );
        check-dup-records ( fail | warn | ignore );
        check-integrity <boolean>;

```

(continues on next page)

(continued from previous page)

```

check-mx ( fail | warn | ignore );
check-mx-cname ( fail | warn | ignore );
check-sibling <boolean>;
check-spf ( warn | ignore );
check-srv-cname ( fail | warn | ignore );
check-wildcard <boolean>;
dialup ( notify | notify-passive | passive | refresh_
↳| <boolean> );
dnssec-dnskey-kskonly <boolean>;
dnssec-loadkeys-interval <integer>;
dnssec-secure-to-insecure <boolean>;
dnssec-update-mode ( maintain | no-resign );
forward ( first | only );
forwarders [ port <port:integer> ] [ dscp
↳<dscp:integer> ] { ( <ipv4_address> | <ipv6_address> ) [
↳port <integer> ] [ dscp <integer> ]; ... };
inline-signing <boolean>;
key-directory <quoted_string>;
max-journal-size <size_no_default>;
max-records <integer>;
max-refresh-time <integer>;
max-retry-time <integer>;
max-transfer-idle-in <integer>;
max-transfer-idle-out <integer>;
max-transfer-time-in <integer>;
max-transfer-time-out <integer>;
max-zone-ttl ( unlimited | <ttlval> );
min-refresh-time <integer>;
min-retry-time <integer>;
multi-master <boolean>;
notify ( explicit | master-only | <boolean> );
notify-delay <integer>;
notify-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
notify-source-v6 ( <ipv6_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
notify-to-soa <boolean>;
nsec3-test-zone <boolean>; // test only
request-ixfr <boolean>;
serial-update-method ( increment | unixtime );
sig-signing-nodes <integer>;
sig-signing-signatures <integer>;
sig-signing-type <integer>;
sig-validity-interval <validity:integer> [ <integer>_
↳];
transfer-source ( <ipv4_address> | * ) [ port (

```

(continues on next page)

(continued from previous page)

```

↪<integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
    try-tcp-refresh <boolean>;
    update-check-ksk <boolean>;
    use-alt-transfer-source <boolean>;
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
};

```

The `options` statement sets up global options to be used by Loop. If there is no `options` statement, an `options` statement with each option set to its default value will be used.

Note

The `options` statement may occur only once in the configuration file.

automatic-interface-scan <boolean>

Automatically rescan network interfaces when the interface addresses are added or removed. This feature uses routing sockets provided by the operating system. The default is `yes`.

avoid-v4-udp-ports { <portrange>; ... }

TBD.

avoid-v6-udp-ports { <portrange>; ... }

TBD.

dnssec-keys-file <quoted_string>

The pathname of a file to override the built-in DNSSEC trust anchors of **named**. It is not configured by default. See the discussion of `dnssec-validation` for details.

named only loads the root zone's trust anchors from the specified file. The file cannot be used to provide trust anchors for other zones. The trust anchors in `dnssec-keys-file` are not used if `dnssec-validation auto` is not in use.

Warning

The use of `dnssec-keys-file` is discouraged. Keeping the Loop software up-to-date is recommended instead, whereby the built-in trust anchors will always be the latest version.

block { <address_match_element>; ... }

Specifies a list of addresses that the server will not accept queries from or use

to resolve a query. Queries from these addresses will be dropped. The default value is none.

coresize (default | unlimited | <sizeval>)

The maximum size of a core dump. The default value is default.

Error

TODO: This description should be updated to describe what default means.

datasize (default | unlimited | <sizeval>)

The maximum amount of data memory the server may use. The default value is default. This is a hard limit on server memory usage. If the server attempts to allocate memory in excess of this limit, the allocation will fail, which may in turn leave the server unable to perform DNS service. Therefore, this option is rarely useful as a way of limiting the amount of memory used by the server, but it can be used to raise an operating system data size limit that is too small by default. If you wish to limit the amount of memory used by the server, use the `max-cache-size` and `recursive-clients` options instead.

Error

TODO: This description should be updated to describe what default means.

directory <quoted_string>

The working directory of the server. The directory specified should be an absolute path. Any non-absolute pathnames in the configuration file will be taken as relative to this directory. The default location for most server output files (e.g. `:named.run`) is this directory. The default value is `.` (the directory from which the server was started).

Note

It is *strongly recommended* that the directory be writable by the effective user ID of the **named** process. For example, unless the `managed-keys-directory` option is configured, the working directory should be writable by **named** so that it can write to the `managed-keys` database.

dscp <integer>

The global DSCP (Differentiated Services Code Point) value to classify outgoing DNS traffic. Valid values are between 0 and 63 inclusive. It is not configured by default.

dump-file <quoted_string>

Pathname of the file that **named** dumps the database to when instructed to do so with the **rndc** `dumpdb` command. The default value is `named_dump.db`.

files (default | unlimited | <sizeval>)

The maximum number of files the server may have open concurrently. The default value is unlimited.

flush-zones-on-shutdown <boolean>

This option controls whether to flush or not flush any pending zone writes when the nameserver exits because it received a SIGTERM signal. The default value is no.

Error

TODO: Check if it applies only due to SIGTERM or to other cases of shutdown too.

heartbeat-interval <integer>

named will perform zone maintenance tasks for all zones marked as dialup whenever this interval expires. The default value is 60 minutes. Reasonable values are up to 1 day (1440 minutes). The maximum value is 28 days (40320 minutes). If set to 0, no zone maintenance for these zones will occur.

Error

TODO: specify units

interface-interval <integer>

named will scan the network interface list every interface-interval minutes. The default value is 60 minutes. The maximum value is 28 days (40320 minutes). If set to 0, interface scanning will only occur when the configuration file is loaded. After the scan, the **named** will begin listening for queries on any newly discovered interfaces (provided they are configured by the listen-on configuration), and will stop listening on interfaces that have gone away.

listen-on [port <port:integer>] [dscp <dscp:integer>] { <acl:address...> }; // may occur multiple times

TBD.

listen-on-v6 [port <port:integer>] [dscp <dscp:integer>] { <acl:address...> }; // may occur multiple times

TBD.

managed-keys-directory <quoted_string>;

Specifies the directory in which **named** stores the managed-keys zone master files. The default value is the working directory (see the directory option).

Note

The directory must be writable by the effective user ID of the **named** process.

If **named** is not configured to use views, then managed keys for the server will be tracked in a single zone master file called `managed-keys.loop`. Otherwise, managed keys will be tracked in separate files, one file per view; each file name will be the SHA-256 hash of the view name, followed by the extension `.mkeys`.

match-mapped-addresses <boolean>;

If enabled, an IPv4-mapped IPv6 address will match any address match elements that match the corresponding IPv4 address. The default value is *no*.

This option was introduced to work around a kernel quirk in some operating systems that causes IPv4 TCP connections, such as zone transfers, to be accepted on an IPv6 socket using mapped addresses. This caused address match lists designed for IPv4 to fail to match. Using this option, **named** solves this problem internally. The use of this option is discouraged.

Error

TODO: Check if this option is required going forward.

max-rsa-exponent-size <integer>;

The maximum RSA public exponent size, in bits, that will be accepted when validating. Valid values are between 17 and 256 inclusive. The default value is 256.

pid-file (<quoted_string> | none);

The pathname of the file the **named** process writes its process ID (PID) to. The default is `/var/run/loop/named.pid`. The PID file can be used by programs that want to send signals to the **named** process. Specifying *none* disables the use of a PID file; no file will be written and any existing file will be removed.

Note

none is a keyword, not a filename, and therefore must not be enclosed in double quotes.

port <integer>;

The UDP/TCP port number the server uses for receiving and sending DNS protocol traffic. The default value is 53.

Note

This option is mainly intended for testing purposes; a nameserver using a port other than 53 will not be able to communicate with the global DNS.

dnssec-accept-expired <boolean>;

If enabled, **named** will accept expired signatures when validating DNSSEC signatures. The TTLs of such expired RRsets is limited to 120 seconds. Setting this option to *yes* leaves **named** vulnerable to replay attacks. The default value is *no*.

querylog <boolean>;

If enabled, **named** logs all queries. The default value is determined by the presence of the logging category *queries*.

recursing-file <quoted_string>;

Pathname of the file that **named** dumps the list of queries that are currently recursing to, when instructed to do so with the **rndc** *recursing* command. The default value is *named.recursing*.

recursive-clients <integer>;

The maximum number of simultaneous resolutions the server will perform on behalf of clients. The default value is 1000.

recursive-clients defines a limit for pending recursive clients. When more clients than this are waiting on resolution, new incoming resolution requests will result in SERVFAIL responses.

reserved-sockets <integer>;

The number of file descriptors reserved for TCP socket descriptors, standard input, etc. This needs to be big enough to cover the number of interfaces **named** listens on, *tcp-clients* as well as to provide room for outgoing TCP queries and incoming zone transfers. The default value is 512. The minimum value for this option is 128, and the maximum value is (*maxsockets* - 128), where *maxsockets* is set by **named** -S.

Note

This option may be removed in the future.

secroots-file <quoted_string>;

Pathname of the file that **named** dumps DNSSEC trust anchors to, when instructed to do so with the **rndc** *secroots* command. The default value is *named.secroot*.

serial-query-rate <integer>;

Slave servers will periodically query master servers to find out if zone serial numbers have changed. Each such query uses a tiny amount of the slave server's network bandwidth. To limit the amount of bandwidth used, Loop limits the rate at which queries are sent to a maximum of the value of the *serial-query-rate* per second. The default value is 20. The minimum value for this option is 1. When set to 0, it is silently increased to 1.

In addition to controlling the rate at which SOA refresh queries are issued, *serial-query-rate* also controls the rate at which NOTIFY messages are sent from both master and slave zones.

server-id (<quoted_string> | none | hostname);

The ID the nameserver should report in the NSID EDNS option in responses, when it receives a query with an NSID (Name Server Identifier) EDNS option (see [RFC 5001](#)). The primary purpose of such queries is to identify which of a group of anycast servers is actually answering queries.

- A value of `none` causes **named** to not return the NSID EDNS option in replies.
- A value of `hostname` causes **named** to return the hostname as found by the `gethostname(2)` function. This is usually equal to the hostname set on the machine.
- In all other cases, the specified value is returned in the NSID EDNS option.

The default value is `none` and no NSID EDNS option is returned in responses.

Note

`none` and `hostname` are keywords, and therefore must not be enclosed in double quotes.

session-keyalg <string>;

The algorithm to use for the TSIG session key. Valid values are *hmac-sha1*, *hmac-sha224*, *hmac-sha256*, *hmac-sha384*, *hmac-sha512*, and *hmac-md5*. The default value is *hmac-sha256*.

Error

TODO: Change this option's type to an enum.

session-keyfile (<quoted_string> | none);

Pathname of the file that **named** writes a generated TSIG session key to, for use by **nsupdate** -l. The default value is `/var/run/loop/session.key`.

Error

TODO: Document the value of `none`.

See the `update-policy` statement's `local` option for more information about this feature.

Note

`none` is a keyword, not a filename, and therefore must not be enclosed in double quotes.

session-keyname <string>;

The key name to use for the TSIG session key. The default value is `local-ddns`.

cookie-algorithm (`hmac-sha256` | `siphash`);

The algorithm used for generating and verifying COOKIE EDNS options. The default value is *siphash*.

cookie-secret <string>;

The secret, encoded as a hex string, used for generating and verifying COOKIE EDNS options. If not set, **named** will generate a random secret at startup. The secret must be 512 bits long for *hmac-sha256* algorithm, and 64 bits long for *siphash* algorithm.

Error

TODO: Change the encoding to Base64 to align with TSIG and control channel HMAC secrets.

stacksize (`default` | `unlimited` | <sizeval>);

The maximum amount of stack memory the nameserver may use. The default is `default`.

Error

TODO: This description should be updated to describe what `default` means.

statistics-file <quoted_string>;

Pathname of the file that **named** appends statistics to, when instructed to do so with the **rndc stats** command. The default value is `named.stats`.

tcp-clients <integer>;

The maximum number of simultaneous client TCP connections that the nameserver will accept. The default value is 100.

tcp-listen-queue <integer>;

The TCP listen queue depth. The default and minimum value is 10. If the kernel supports the "dataready" *accept(2)* filter, this also controls the number of TCP connections that will be queued in kernel space waiting for some data before being passed to *accept(2)*.

- Non-zero values less than 10 will be silently raised to 10.
- A value of 0 may also be used. On most platforms this sets the TCP listen queue depth to a system-defined default value.

tkey-dhkey <name:quoted_string> <keyid:integer>;

The Diffie-Hellman key used by the server to generate shared keys with clients using the Diffie-Hellman mode of TKEY. It is not configured by default. The nameserver must be able to load the public and private keys from files in the

working directory. In most cases, the key name should be the nameserver's host-name.

tkey-domain <quoted_string>;

The domain name appended to the names of all shared keys generated with TKEY. It is not configured by default.

When a client requests a TKEY exchange, it may or may not specify the desired name for the key. If present, the name of the shared key will be:

```
client specified part + tkey-domain
```

Otherwise, the name of the shared key will be:

```
random hex digits + tkey-domain
```

In most cases, the tkey-domain's value should be the nameserver's domain name, or an otherwise non-existent sub-domain like `_tkey.<domain_name>`.

Note

If you are using GSS-TSIG, this variable must be defined, unless you specify a specific keytab using `tkey-gssapi-keytab`.

tkey-gssapi-credential <quoted_string>;

The security credential with which the server should authenticate keys requested by the GSS-TSIG protocol. It is not configured by default. Currently only Kerberos 5 authentication is available and the credential is a Kerberos principal which the server can acquire through the default system Kerberos keytab file, typically `/etc/krb5.keytab`. The location of the keytab file can be overridden using the `tkey-gssapi-keytab` option. Normally this principal is of the form `DNS/<host.domain>`.

Note

To use GSS-TSIG, `tkey-domain` must also be set if a specific keytab is not set with `tkey-gssapi-keytab`.

tkey-gssapi-keytab <quoted_string>;

Pathname of the Kerberos 5 keytab file to use for GSS-TSIG updates. It is not configured by default. If this option is configured and `tkey-gssapi-credential` is not configured, then transactions will be allowed with any key matching a principal in the keytab.

transfers-in <integer>;

The maximum number of inbound zone transfers that can occur concurrently. The default value is 10. Increasing `transfers-in` may speed up the convergence of slave zones, but it also may increase the load on the local machine.

transfers-out <integer>;

The maximum number of outbound zone transfers that can occur concurrently. Zone transfer requests in excess of the limit will be refused. The default value is 10.

transfers-per-ns <integer>;

The maximum number of inbound zone transfers that can be concurrently transferring from any single remote nameserver. The default value is 2. Increasing `transfers-per-ns` may speed up the convergence of slave zones, but it also may increase the load on the remote nameserver. `transfers-per-ns` may be overridden on a per-nameserver basis by using the `transfers` phrase of the `server` statement.

use-v4-udp-ports { <portrange>; ... };

TBD.

use-v6-udp-ports { <portrange>; ... };

TBD.

version (<quoted_string> | none);

The version the nameserver should report in response to a query of the name `version.loop` with type `TXT`, and class `CHAOS`. The default is the real version number of this server. Specifying a value of `none` disables processing of the queries.

Note

`none` is a keyword, and therefore must not be enclosed in double quotes.

Error

TODO: "disables processing of the queries" should be described better.

allow-new-zones <boolean>;

If enabled, zones can be dynamically added at runtime via the `rndc addzone` command, and such dynamically added zones can be deleted via the `rndc delzone` command. The default value is `no`.

allow-query-cache { <address_match_element>; ... };

Specifies which hosts are allowed to query answers from the cache. If `allow-query-cache` is not set then the value of `allow-recursion` is used if set, otherwise the value of `allow-query` is used if set unless `recursion no`; is set in which case `none` is used, otherwise the default (`localnets`; `localhost`;) is used.

allow-query-cache-on { <address_match_element>; ... };

Specifies local addresses to which queries made can give answers from the cache.

If not specified, the default is to allow cache queries on any address, `localnets` and `localhost`.

allow-recursion { <address_match_element>; ... };

Specifies which hosts are allowed to make recursive queries through this name-server. If `allow-recursion` is not set then `allow-query-cache` is used if set, otherwise `allow-query` is used if set, otherwise the default (`localnets; localhost;`) is used.

allow-recursion-on { <address_match_element>; ... };

Specifies which local addresses can accept recursive queries. The default is to allow recursive queries on all addresses.

attach-cache <string>;

Allows multiple views to share a single cache database. Each view has its own cache database by default, but if multiple views have the same operational policy for name resolution and caching, those views can share a single cache to save memory and possibly improve cache hit rate and resolution efficiency by using this option.

The `attach-cache` option may also be specified in `view` statements, in which case it overrides the global `attach-cache` option.

The option's value specifies the cache name to be shared. When **named** configures `view` statements which are supposed to share a cache, it creates a cache with the specified name for the first view of these sharing views. The rest of the views will simply refer to the already created cache.

One common configuration to share a cache would be to allow all views to share a single cache. This can be done by specifying the `attach-cache` as a global option with an arbitrary name.

Another possible operation is to allow a subset of all views to share a cache while the others to retain their own caches. For example, if there are three views A, B, and C, and only A and B should share a cache, specify the `attach-cache` option as a view A (or B)'s option, referring to the other view name:

```
view "A" {
    // this view has its own cache
    ...
};

view "B" {
    // this view refers to A's cache
    attach-cache "A";
};

view "C" {
    // this view has its own cache
    ...
};
```

Views that share a cache must have the same policy on configurable parameters that may affect caching. The current implementation requires the following configurable options be consistent among these views: `check-names`, `dnssec-validation`, `max-cache-ttl`, `max-ncache-ttl`, `max-cache-size`, and `zero-no-soa-ttl`.

Error

TODO: Check if this is enforced in code.

Note

There may be other parameters that may cause confusion if they are inconsistent for different views that share a single cache. For example, if these views define different sets of forwarders that can return different answers for the same question, sharing the answer does not make sense or could even be harmful. It is administrator's responsibility to ensure configuration differences in different views do not cause disruption with a shared cache.

auth-nxdomain <boolean>; // default changed

If enabled, the AA bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default value is `no`. If you are using very old DNS software, you may need to set it to `yes`.

cache-file <quoted_string>; // test only

Pathname of the file that **named** loads the initial contents of the resolver's cache from. It is not configured by default.

Warning

This option is present for testing only. It is only of interest to Loop developers and may be removed or changed in a future release. Please don't use it.

check-names (master | slave | response) (fail | warn | ignore);
// may occur multiple times

This option is used to restrict the character set and syntax of certain domain names in master files and/or DNS responses received from the network. The default varies according to usage area:

- For master zones the default is `fail`.
- For slave zones the default is `warn`.
- For answers received from the network (`response`) the default is `ignore`.

The rules for legal hostnames and mail domains are derived from [RFC 952](#), and [RFC 821](#) as modified by [RFC 1123](#).

`check-names` applies to the owner names of A, AAAA, and MX records. It also applies to the domain names in the RDATA of NS, SOA, MX, and SRV records. It also applies to the RDATA of PTR records where the owner name indicated that it is a reverse lookup of an IP address (the owner name ends in IN-ADDR.ARPA, IP6.ARPA, or IP6.INT).

clients-per-query <integer>;

Sets the initial and minimum number of concurrent recursive clients for any given query (<qname, qtype, qclass>) that **named** will accept before dropping additional clients. The default value is 10. **named** will attempt to self-tune the number of concurrent clients using `clients-per-query`'s value as the minimum and `max-clients-per-query`'s value as the maximum, and changes will be logged.

This value should reflect the number of queries that come in for a given name during the time it takes to resolve that name. If the number of queries exceed this value, **named** will assume that it is dealing with a non-responsive zone and will drop the following queries. If it gets a response after dropping queries, it will raise the estimate. The estimate will then be lowered in 20 minutes if it has remained unchanged.

If `clients-per-query` is set to 0, then there is no limit on the number of clients per query and no queries will be dropped.

deny-answer-addresses { <acl:address_match_element>; ...
} [except-from { <except-from:quoted_string>; ... }];

TBD.

deny-answer-aliases { <name:quoted_string>; ...
} [except-from { <except-from:quoted_string>; ... }];

TBD.

disable-algorithms <domain:string> { <algorithms:string>; ...
}; // may occur multiple times

Disable the specified DNSSEC algorithms at and below the specified domain. This option can be specified multiple times. It is not configured by default.

The longest domain-matched `disable-algorithms` clause will be used to determine which algorithms are used. If all supported algorithms are disabled, the zones covered by the `disable-algorithms` will be treated as insecure.

disable-ds-digests <domain:string> { <digests:string>; ... };
// may occur multiple times

Disable the specified DS digest types at and below the specified domain. This option can be specified multiple times. It is not configured by default.

The longest domain-matched `disable-ds-digests` clause will be used to determine which digest types are used. If all supported digest types are disabled, the zones covered by the `disable-ds-digests` will be treated as insecure.

disable-empty-zone <string>; // may occur multiple times

Disable individual empty zones. This option can be specified multiple times. It

is not configured by default. See the discussion of empty zones in the Loop User Manual.

dns64 <netprefix> { ... };

```
dns64 <netprefix> {
    break-dnssec <boolean>;
    clients { <address_match_element>; ... };
    exclude { <address_match_element>; ... };
    mapped { <address_match_element>; ... };
    recursive-only <boolean>;
    suffix <ipv6_address>;
}; // may occur multiple times
```

This directive instructs **named** to return mapped IPv4 addresses in answers to AAAA queries when there are no AAAA records. It is intended to be used in conjunction with NAT64. Each **dns64** option defines one DNS64 prefix. Multiple DNS64 prefixes can be defined. It is not configured by default.

Compatible IPv6 prefixes have lengths of 32, 40, 48, 56, 64 and 96 as per [RFC 6052](#).

Additionally a reverse IP6.ARPA zone will be created for the prefix to provide a mapping from the IP6.ARPA names to the corresponding IN-ADDR.ARPA names using synthesized CNAMEs. The **dns64-server** and **dns64-contact** options can be used to specify the SOA MNAME (primary server) and SOA RNAME (responsible person's mailbox) respectively for the zones.

Each **dns64** option supports the following optional sub-clauses:

- The **clients** ACL determines which clients are affected by this directive. The default value is **any**.
- The **mapped** ACL selects which IPv4 addresses are to be mapped in the corresponding A RRset. The default value is **any**.
- Normally, DNS64 won't apply to a domain name that owns one or more AAAA records; these records will simply be returned. The **exclude** ACL allows specification of a list of IPv6 addresses that will be ignored if they appear in a domain name's AAAA records, and DNS64 will be applied to any A records the domain name owns. The default value is **::ffff:0.0.0.0/96**.
- **suffix** can be defined to set the bits trailing the mapped IPv4 address bits. The bits matching the prefix and mapped IPv4 address must all be 0. The default value is **::**.
- If **recursive-only** is set to **yes** the DNS64 synthesis will only happen for recursive queries. The default value is **no**.
- If **break-dnssec** is set to **yes** the DNS64 synthesis will happen even if the result, if validated, would cause a DNSSEC validation failure. If this option is set to **no**, and if DO=1 in the incoming query, and there are RRSIGs on the applicable records, then synthesis will not happen. The default value is **no**.

dns64-contact <string>;

TBD.

dns64-server <string>;

TBD.

dnssec-enable <boolean>;

Whether DNSSEC-related resource records are to be returned by **named**. If set to **no**, **named** will not return DNSSEC-related resource records unless specifically queried for. The default value is **yes**.

dnssec-must-be-secure <domain:string> <value:boolean>; //

may occur multiple times

Specify domains which must be or may not be secure (signed and validated). If set to **yes**, then **named** will only accept answers if they are secure at and below the specified domain. If **no**, then normal DNSSEC validation applies allowing for insecure answers to be accepted. The specified domain must be under a **trusted-keys** or **managed-keys** statement, or **dnssec-validation auto** must be active. This option can be specified multiple times. It is not configured by default.

dnssec-validation (**yes** | **no** | **auto**);

Whether DNSSEC validation should be enabled.

- If set to **no**, DNSSEC validation is disabled.
- If set to **yes**, DNSSEC validation is enabled, but a trust anchor should be explicitly configured using a **trusted-keys** or **managed-keys** statement. If no trust anchors are explicitly configured, then the root will be treated as insecure.
- If set to **auto**, DNSSEC validation is enabled, and ICANN root zone trust anchors that are built into **named** are used. The built-in trust anchors are current as of the software release date. If they expire, updated trust anchors may be provided using the **dnssec-keys-file** config option.

The default value is **yes**.

Note

dnssec-enable also needs to be set to **yes** for this option to be effective.

Note

Whenever the resolver sends queries to an EDNS-compliant server, it always sets the DO bit indicating it can support DNSSEC responses, even if **dnssec-validation** is off.

dual-stack-servers [port <port:integer>] { (<quoted_string> [port <in... >]);

Specifies hostnames or addresses of machines with access to both IPv4 and IPv6 transports. If a hostname is used, the nameserver must be able to resolve the name using only the transport it has. If the machine is dual stacked, then the `dual-stack-servers` clause has no effect unless access to a transport has been disabled on the command line (e.g. with the **named** -4 argument).

Error

TODO: This has to be explained in more detail.

edns-udp-size <integer>;

Sets the maximum advertised EDNS requestor's UDP payload size in octets (see [RFC 6891](#) section 6.2.3) that is advertised by **named** when querying remote nameservers. Values should be configured between 512 and 4096 inclusive; values outside this range will be silently adjusted to the nearest value within it. The default value is 4096.

The usual reason for setting this option to a non-default value is to get DNS over UDP answers to pass through broken firewalls that block fragmented packets and/or block UDP DNS packets that are greater than 512 bytes.

When **named** first queries a remote server, it will advertise a UDP buffer size of 512, as this has the greatest chance of success on the first try:

- If the initial response times out, **named** will try again with plain DNS, and if that is successful, it will be taken as evidence that the server does not support EDNS. After some failures using EDNS and successes using plain DNS, **named** will default to plain DNS for future communications with that server. Periodically, **named** will attempt an EDNS query to see if the situation has improved.
- If the initial query was successful with the UDP buffer size of 512, then **named** will advertise progressively larger buffer sizes on successive queries, until responses begin timing out or the maximum (value of this option) is reached.

The default UDP buffer sizes advertised by **named** are 512, 1232, 1432, and 4096, but never exceeding this option's value. The values 1232 and 1432 are chosen to allow for an IPv4/IPv6 encapsulated UDP message to be sent without fragmentation at the minimum MTU sizes for Ethernet and IPv6 networks.

empty-contact <string>;

Specify what value will appear in the returned SOA record's RNAME field for empty zones. The default value is "." (the root name). See the discussion of empty zones in the Loop User Manual.

empty-server <string>;

Specify what value will appear in the returned SOA record's MNAME field for

empty zones. If none is specified, then the zone's name will be used. See the discussion of empty zones in the Loop User Manual.

empty-zones-enable <boolean>;

Whether empty zones should be enabled. The default is `yes`. See the discussion of empty zones in the Loop User Manual.

fetch-quota-params <frequency:integer> <low:fixedpoint> <high:fixedpoint>

Sets the parameters to use for dynamic resizing of the `fetches-per-server` quota in response to detected congestion.

- The *frequency* value indicates how frequently to recalculate the moving average of the ratio of timeouts to responses for each server. The default value is 100, meaning we recalculate the average ratio after every 100 queries have either been answered or timed out.
- The *low* value indicates the "low" threshold of timeout ratio. The default value is 0.1.
- The *high* value indicates the "high" threshold of timeout ratio. The default value is 0.3.
- The *discount* value indicates the discount rate for the moving average. A higher discount rate causes recent events to weigh more heavily when calculating the moving average; a lower discount rate causes past events to weigh more heavily, smoothing out short-term blips in the timeout ratio. The default value is 0.7.

The *low*, *high*, and *discount* fields accept values of fixed-point type with precision of 1/100, i.e., at most two places after the decimal point are significant.

Error

TODO: Improve this description; explain `fetches-per-server` in the features section of the Loop User Manual.

fetches-per-server <fetches:integer> [(drop | fail)];

The maximum number of concurrent iterative queries that the resolver will allow to be sent to a single upstream nameserver before blocking additional queries. The value of *fetches* should reflect how many fetches would normally be sent to any one server in the time it would take to resolve them. It should be lower than `recursive-clients`. The default value is 0, i.e., there is no limit on the number of fetches per server and no queries will be dropped or answered with SERVFAIL.

Optionally, the value may be followed by the keyword `drop` or `fail`, indicating whether queries will be dropped with no response, or answered with SERVFAIL, when all of the nameservers authoritative for a zone are found to have exceeded the per-server quota. The default is `fail`.

The `fetches-per-server` quota is dynamically adjusted in response to detected congestion. As resolver queries are sent to an upstream nameserver, and are either answered or timeout, an exponentially weighted moving average is calculated of the ratio of timeouts to responses:

- If the current average timeout ratio rises above a *high* threshold, then `fetches-per-server` is reduced for that server.
- If the current average timeout ratio drops below a *low* threshold, then `fetches-per-server` is increased.

The `fetch-quota-params` option can be used to adjust the parameters for this calculation.

fetches-per-zone <fetches:integer> [(drop | fail)];

The maximum number of concurrent iterative queries to any one domain that the server will permit before blocking new queries for data in or beneath that zone. This value should reflect how many fetches would normally be sent to any one zone in the time it would take to resolve them. It should be smaller than `recursive-clients`. The default value is 0, i.e., there is no limit on the number of fetches per zone and no queries will be dropped or answered with SERVFAIL.

When many clients simultaneously query for the same name and type, the clients will all be attached to the same fetch, up to the `max-clients-per-query` limit, and only one iterative query will be sent. However, when clients are simultaneously querying for *different* names or types, multiple queries will be sent and `max-clients-per-query` is not effective as a limit.

Optionally, the value may be followed by the keyword `drop` or `fail`, indicating whether queries will be dropped with no response, or answered with SERVFAIL, when queries exceed the fetch quota for a zone. The default is `drop`.

The current list of active fetches can be dumped by running the **rndc** `recurring` command to a file named `named.recurring` in the working directory. The list includes the number of active fetches for each domain and the number of queries that have been passed or dropped as a result of the `fetches-per-zone` limit.

Note

These counters are not cumulative over time; whenever the number of active fetches for a domain drops to zero, the counter for that domain is deleted, and the next time a fetch is sent to that domain, it is recreated with the counters set to zero.

Error

TODO: Should this option be renamed to `fetches-per-domain`?

ixfr-from-differences (master | slave | <boolean>);

If enabled, whenever the server loads a new version of a master zone from its zone file or receives a new version of a slave zone via zone transfer, it will compare the new version to the previous version and calculate a set of differences. The differences are then logged in the zone's journal file such that the changes can be transmitted to downstream slaves as an IXFR (incremental zone transfer). The default value is `no`.

By allowing IXFRs to be used for non-dynamic zones, this option saves bandwidth at the expense of increased CPU and memory consumption on the master's nameserver. In particular, if the new version of a zone is completely different from the previous one, the set of differences will be of a size comparable to the combined size of the old and new zone versions, and the nameserver will need to temporarily allocate memory to hold this complete difference set.

`ixfr-from-differences` also accepts `master` and `slave` at the `view` and `options` statement levels which causes `ixfr-from-differences` to be enabled for all `master` or `slave` zones respectively.

max-cache-size <size_no_default>;

The maximum amount of memory in bytes to use for the resolver's cache. When the amount of data in the cache reaches this limit, the resolver will evict cached records prematurely based on an LRU strategy so that the limit is not exceeded. The keyword `unlimited` or the value `0` will place no limit on cache size; records will be purged from the cache only when their TTLs expire. Any positive values less than 2MB will be silently increased to 2MB. When there are multiple views, the limit applies separately to the cache of each view. The default value is `unlimited`.

Error

TODO: Add a keyword for automatic maximum size computation based on the machine's available RAM.

max-cache-ttl <integer>;

Sets the maximum TTL in seconds for which the nameserver will cache positive answers. The default value is 7 days. A value of 0 may cause all queries to return SERVFAIL, because of lost caches of intermediate RRsets (such as NS and glue AAAA/A records) in the resolution process.

Error

TODO: In this case, a value of 0 is not useful and there should be a sane minimum.

max-clients-per-query <integer>;

Sets the maximum number of concurrent recursive clients for any given query (<qname, qtype, qclass>) that **named** will accept before dropping additional

clients. See `clients-per-query` for more details. The default value is 100. If the value is set to 0, then there is no upper bound other than imposed by `recursive-clients`.

max-ncache-ttl <integer>;

To reduce work, reduce network traffic, and increase performance, **named** also caches negative answers (see [RFC 2308](#)). This option sets the maximum TTL in seconds for which the nameserver will cache negative answers. The value cannot exceed 7 days and will be silently truncated to 7 days if set to a greater value. The default value is 3 hours.

max-recursion-depth <integer>;

Sets the maximum number of levels of recursion that are permitted at any one time while servicing a recursive query. Resolving a name may require looking up nameserver addresses, which in turn requires resolving hostnames, etc. If the number of indirections exceeds this value, the recursive query is terminated and returns SERVFAIL. The default value is 7.

max-recursion-queries <integer>;

Sets the maximum number of iterative queries that may be sent while servicing a recursive query. If more queries are sent, the recursive query is terminated and returns SERVFAIL. Queries to look up top-level domains such as `com.` and `net.` and the DNS root zone are exempt from this limitation. The default value is 75.

max-udp-size <integer>;

Sets the maximum EDNS UDP response message size that **named** will send in bytes. Values should be configured between 512 and 4096 inclusive; values outside this range will be silently adjusted to the nearest value within it. The default value is 4096.

The usual reason for setting this option to a non-default value is to get UDP answers to pass through broken firewalls that block fragmented packets and/or block UDP packets that are greater than 512 bytes.

This value applies to responses sent by **named**. To set the maximum advertised EDNS requestor's UDP payload size in queries, see `edns-udp-size`.

Note

Setting this to a low value could result in truncated DNS over UDP answers, and cause additional DNS over TCP traffic to the nameserver.

minimal-responses <boolean>;

If enabled, when generating responses the server will only add records to the authority and additional data sections when they are required (delegations, negative responses, etc.). Enabling this option may reduce the amount of processing necessary to respond to queries, and improve the performance of the nameserver. The default value is `no`.

no-case-compress { <address_match_element>; ... };

Specifies a list of addresses for which **named** should perform case-sensitive name compression in response messages. This ACL can be used when **named** answers to clients that do not comply with the requirement in [RFC 1035](#) to use case-insensitive name comparisons when checking for matching domain names.

The default value is `none`, i.e., case-insensitive compression will be used for all clients.

Case-insensitive compression can result in slightly smaller responses: if a response contains RRs with owner names `example.com` and `example.COM`, case-insensitive compression would treat the second one as a duplicate. It also ensures that the case of all the owner names of returned RRs exactly matches the case of the query name in the question section, rather than matching the case of the records entered in the zone file.

Case-insensitive compression is *always* used in AXFR and IXFR responses, regardless of whether the client matches this ACL.

There are circumstances in which **named** will not preserve the case of owner names of records. If a zone master file contains records of different types with the same name, but the capitalization of the name is different (e.g., `"www.example.com./A"` and `"WWW.EXAMPLE.COM./AAAA"`), then all responses for that name will use the *first* version of the name that was used in the zone file. However, domain names specified in the RDATA of resource records (i.e., records of type NS, MX, CNAME, etc) will always have their case preserved unless the client matches this ACL.

Error

TODO: Explain this some more, and in the Loop User Manual as well.

nocookie-udp-size <integer>;

Sets the maximum size in bytes of UDP responses that will be sent to queries without a valid COOKIE EDNS option. A value below 128 will be silently increased to 128. The default value is 4096, but the `max-udp-size` option may further limit the response size.

preferred-glue <string>;

If specified, the listed type (A or AAAA) will be emitted before other glue in the additional section of a query response. The default is to prefer A records when responding to queries that arrived via IPv4, and AAAA when responding to queries that arrived via IPv6.

Error

TODO: Change to an enum type.

prefetch <trigger:integer> [<integer>];

When a query is received for cached data which is to expire shortly, **named** can refresh the data from the authoritative server immediately, ensuring that the cache has an answer available for future queries.

The *trigger* value specifies the TTL in seconds at which prefetch of the current query will take place. When a cached record with a lower TTL value is encountered during query processing, it will be refreshed. *trigger* values should be configured between 1 and 10. Values larger than 10 will be silently reduced to 10. Setting *trigger* to 0 causes the prefetching to be disabled. The default value for *trigger* is 2.

An optional second argument specifies the *eligibility* TTL value in seconds. The *eligibility* TTL is the smallest *original* TTL value of an RRset that will be accepted for prefetching. The eligibility TTL must be at least 6 seconds higher than the trigger TTL; if it isn't, **named** will silently increase it to be 6 seconds higher. The default value for the eligibility TTL is 9.

provide-ixfr <boolean>;

Whether **named**, when acting as master, will respond with IXFRs (incremental zone transfers; see [RFC 1995](#)) when a remote nameserver, a slave or client, requests it.

- If set to *yes*, incremental transfer will be provided whenever possible.
- If set to *no*, all transfers to the remote server or client will be AXFRs (non-incremental full zone transfer; see [RFC 5936](#)).

The default value is *yes*.

query-source (([address] (<ipv4_address> | *) [port (<integer> |

The *query-source* clause specifies the IPv4 source address to be used for DNS queries sent to remote nameservers.

Error

TODO: Expand this description with the text from the "Query address" section.

query-source-v6 (([address] (<ipv6_address> | *) [port (<integer>

The *query-source-v6* clause specifies the IPv6 source address to be used for DNS queries sent to remote nameservers.

Error

TODO: Expand this description with the text from the "Query address" section.

rate-limit { ... };

```
rate-limit {
    all-per-second <integer>;
    errors-per-second <integer>;
    exempt-clients { <address_match_element>; ... };
    ipv4-prefix-length <integer>;
    ipv6-prefix-length <integer>;
    log-only <boolean>;
    max-table-size <integer>;
    min-table-size <integer>;
    nodata-per-second <integer>;
    nxdomains-per-second <integer>;
    qps-scale <integer>;
    referrals-per-second <integer>;
    responses-per-second <integer>;
    slip <integer>;
    window <integer>;
};
```

TBD.

recursion <boolean>;

If enabled and if a DNS query requests recursion (RD=1), then the server will attempt to perform any recursive resolution required to answer the query. If the option is disabled and the server does not already know the answer, it will return a referral response. The default value is `yes`.

Note

Disabling this option does not prevent clients from getting data from the nameserver's cache. It only prevents new recursive resolutions from being performed for client queries. Recursive resolution and caching may still occur as a side-effect the nameserver's internal operation, such as NOTIFY address lookups.

request-nsid <boolean>;

If enabled, an empty NSID (Name Server Identifier) EDNS option (see [RFC 5001](#)) is sent in all queries to name servers during iterative resolution, and also in SOA queries performed during zone refreshes. During resolution, if the name server returns an NSID EDNS option in its response, then its contents are logged in the `resolver` category at level `info`. The default value is `no`.

Error

TODO: NSID is sent in NS queries too in the zonemgr. Also, should NSID from SOA refresh's responses be logged too?

request-cookie <boolean>;

If enabled, **named** adds a COOKIE EDNS option (see [RFC 7873](#)) in requests sent to remote nameservers. The default value is `yes`.

If **named** determines that the COOKIE EDNS option is not supported by some remote nameservers, it may not add COOKIE EDNS options in requests to them.

If the resolver has previously talked to the server, the cookie returned in the previous transaction is sent to it. This is used by the server to determine whether the resolver has talked to it previously. A resolver sending the correct cookie is usually assumed not to be an off-path attacker sending a spoofed-source query; the query is therefore unlikely to be part of a reflection/amplification attack, so resolvers sending a correct COOKIE EDNS option are usually not subject to response rate limiting (RRL). Resolvers which do not send a correct COOKIE EDNS option may be limited to receiving smaller responses via the `nocookie-udp-size` option.

Error

TODO: Should cookies be sent by the zonemgr too in the requests it makes?

resolver-query-timeout <integer>;

The amount of time in seconds that the resolver will spend attempting to resolve a recursive query before failing. The default value is 10. Values should be configured between 10 and 30. If set to 0, it will be silently increased to the default value.

response-policy { zone <quoted_string> [policy (cname | disabled | drop
...
] [recursive-only <boolean>] [break-dnssec <boolean>] [max-policy-tt

TBD.

root-delegation-only [exclude { <quoted_string>; ... }];

Turn on enforcement of `delegation-only` in TLDs (top level domains) and root zones with an optional exclude list. It is not configured by default.

DS queries are expected to be made to and be answered by delegation-only zones. Such queries and responses are treated as an exception to delegation-only processing and are not converted to NXDOMAIN responses provided a CNAME is not discovered at the query name.

If a delegation-only zone's name server also serves its child zone, it is not always possible to determine whether an answer comes from the delegation-only zone or the child zone. SOA, NS, and DNSKEY records are apex-only records and a matching response that contains these records or DS is treated as coming from a child zone. RRSIG records are also examined to see if they are signed by a child zone or not. The authority section is also examined to see if there is evidence that the answer is from the child zone. Answers that are determined to be from a

child zone are not converted to NXDOMAIN responses. Despite all these checks, there is still a possibility of false negatives when a child zone is being served.

Similarly false positives can arise from empty nodes (no records at the name) in the delegation-only zone when the query type is not ANY.

Note

Some TLDs are not delegation-only. E.g. "DE", "LV", "US", "MUSEUM", etc. As an example, they can be configured as:

```
options {  
    root-delegation-only exclude { "de"; "lv"; "us";  
    ↪ "museum"; };  
};
```

rrset-order (none | random);

Configures the order of records in RRsets returned in answers:

- If the value is `random`, the records are re-ordered to be in random order.
- If the value is `none`, no re-ordering is performed.

The default value is `none`.

zero-no-soa-ttl-cache <boolean>;

When caching a negative response to a SOA query, set the TTL to 0. The default value is `no`.

allow-notify { <address_match_element>; ... };

Configures which hosts are allowed to send DNS NOTIFY (see [RFC 1996](#)) messages to this nameserver of zone changes, in addition to the zone's masters. It is only meaningful for a slave zone. The default value is `none`.

allow-query { <address_match_element>; ... };

Configures which hosts are allowed to send ordinary DNS queries to this nameserver. The default value is `any`.

allow-query-on { <address_match_element>; ... };

Configures which local addresses can receive ordinary DNS queries. This makes it possible, for instance, to receive queries on internal-facing interfaces, but disallow them on external-facing ones, without necessarily knowing the internal network's addresses. The default value is `any`.

Note

`allow-query-on` is only checked for queries that are permitted by `allow-query`. A query must be allowed by both ACLs, or it will be refused.

allow-transfer { <address_match_element>; ... };

Configures which addresses are allowed to transfer zones from this nameserver. The default value is any.

allow-update { <address_match_element>; ... };

Configures which hosts are allowed to perform DNS UPDATE transactions for master zones. If it is not configured, updates are refused. It is not configured by default.

Warning

Allowing updates based on the requestor's IP address is insecure, as the source IP address may be spoofed in the case UDP. See the Loop User Manual for more details.

allow-update-forwarding { <address_match_element>; ... };

Specifies which hosts are allowed to submit DNS UPDATES to slave zones to be forwarded to the master. The default value is none, which means that no update forwarding will be performed. To enable update forwarding, specify a value of any. Specifying values other than none or any is usually counter-productive, as the responsibility for DNS UPDATE access-control should rest with the master server, not the slave servers.

Warning

Allowing update forwarding on a slave server may expose master servers relying on insecure IP address based access-control to attacks. See the Loop User Manual for more details.

also-notify [port <port:integer>] [dscp <dscp:integer>] { (<masters> ...);

Configures a list of IP addresses of nameservers that are also sent DNS NOTIFY messages whenever a fresh copy of the zone is loaded, in addition to the servers listed in the zone's NS records. This helps to ensure that copies of the zones will quickly converge on stealth nameservers. It is not configured by default.

Optionally, a port may be specified with each also-notify address to send the notify messages to a port other than the default of 53. An optional TSIG key can also be specified with each address to cause the notify messages to be signed; this can be useful when sending notifies to multiple views. In place of explicit addresses, one or more named masters lists can be used.

Note

If a zone's notify option is set to no, the IP addresses in the also-notify list will not be sent DNS NOTIFY messages for that zone.

alt-transfer-source (<ipv4_address> | *) [port (<integer> | *)] [d

Configures an alternate transfer source IPv4 address, for use if the one listed in `transfer-source` fails and `use-alt-transfer-source` is set.

Note

If you do not wish the alternate transfer source to be used, you should set `use-alt-transfer-source` appropriately and you should not depend upon getting an answer back to the first refresh query.

alt-transfer-source-v6 (<ipv6_address> | *) [port (<integer> | *)]

Configures an alternate transfer source IPv6 address, for use if the one listed in `transfer-source-v6` fails and `use-alt-transfer-source` is set.

Note

If you do not wish the alternate transfer source to be used, you should set `use-alt-transfer-source` appropriately and you should not depend upon getting an answer back to the first refresh query.

auto-dnssec (allow | maintain | off);

Zones configured for dynamic DNS may use this option to allow varying levels of automatic DNSSEC key management. There are three possible values:

- `allow` permits keys to be updated and the zone fully re-signed whenever the user issues the **rndc** `sign` command.
- `maintain` includes the above, but also automatically adjusts the zone's DNSSEC keys on schedule, according to the keys' timing metadata (see `dnssec-keygen(1)` and `dnssec-settime(1)`).

The **rndc** `sign` command causes **named** to load keys from the key repository and sign the zone with all keys that are active.

The **rndc** `loadkeys` command causes **named** to load keys from the key repository and schedule key maintenance events to occur in the future, but it does not sign the full zone immediately.

Note

Once keys have been loaded for a zone the first time, the repository will be searched for changes periodically, regardless of whether the **rndc** `loadkeys` command is used. The recheck interval is defined by `dnssec-loadkeys-interval`.

The default value is `off`.

Error

TODO: This description has to be rewritten.

check-dup-records (fail | warn | ignore);

Check master zones for records that are treated as different by DNSSEC but are semantically equal in plain DNS. The default value is `warn`.

Error

TODO: This description has to be rewritten.

check-integrity <boolean>;

Whether to perform post-load zone integrity checks on master zones. This checks that MX and SRV records refer to address (A or AAAA) records and that glue address records exist for delegated zones. For MX and SRV records only in-zone hostnames are checked (for out-of-zone hostnames use **named-checkzone**). For NS records only names below top of zone are checked (for out-of-zone names and glue consistency checks use **named-checkzone**). The default value is `yes`.

The use of type SPF records for publishing "Sender Policy Framework" is deprecated as the migration from using type TXT records to type SPF records was abandoned. Enabling this option also checks that a type TXT "Sender Policy Framework" record exists (starts with "v=spf1") if there is an SPF record. Warnings are emitted if the type TXT record does not exist and can be suppressed with `check-spf`.

check-mx (fail | warn | ignore);

Check whether the MX record appears to refer to a IP address. The default value is `warn`.

check-mx-cname (fail | warn | ignore);

If `check-integrity` is enabled, then fail, warn, or ignore MX records that refer to CNAMEs. The default value `warn`.

check-sibling <boolean>;

When performing integrity checks, also check that sibling glue exists. The default value is `yes`.

check-spf (warn | ignore);

If `check-integrity` is enabled, then check that there is a type TXT "Sender Policy Framework" record present (starts with "v=spf1") if there is a type SPF record present. The default value is `warn`.

check-srv-cname (fail | warn | ignore);

If `check-integrity` is enabled, then fail, warn, or ignore SRV records that refer to CNAMEs. The default value is warn.

check-wildcard <boolean>;

If enabled, check for non-terminal wildcard owner names and issue a warning if they are present. The use of non-terminal wildcard owner names is usually due to a mis-understanding of the wildcard matching algorithm (see [RFC 1034](#)). This option affects master zones. The default value is yes.

dialup (notify | notify-passive | passive | refresh | <boolean>);

If set to yes, then the server treats all zones as if they are doing zone transfers across a dial-on-demand dialup link, which can be brought up by traffic originating from this server. This has different effects according to zone type and concentrates the zone maintenance so that it all happens in a short interval, once every `heartbeat-interval` and hopefully during the one call. It also suppresses some of the normal zone maintenance traffic. The default value is no.

If the zone is a master zone, then the server will send out a NOTIFY request to all the slaves (default). This should trigger the zone SOA serial number check in the slave (providing it supports NOTIFY) allowing the slave to verify the zone while the connection is active. The set of servers to which NOTIFY is sent can be controlled by the `notify` and `also-notify` config options.

If the zone is a slave or stub zone, then the server will suppress the regular "zone up to date" (refresh) queries and only perform them when the `heartbeat-interval` expires in addition to sending NOTIFY requests.

Finer control can be achieved by using the values:

- `notify` which only sends NOTIFY messages,
- `notify-passive` which sends NOTIFY messages and suppresses the normal refresh queries,
- `refresh` which suppresses normal refresh processing and sends refresh queries when the `heartbeat-interval` expires, and
- `passive` which just disables normal refresh processing.

Value	Normal refresh	Heartbeat refresh	Heartbeat notify
no (default)	yes	no	no
yes	no	yes	yes
notify	yes	no	yes
refresh	no	yes	no
passive	no	no	no
notify-passive	no	no	yes

Note

Normal DNS NOTIFY processing is not affected by the `diagnose` option.

dnssec-dnskey-kskonly <boolean>;

When this option and the `update-check-ksk` option's value are both enabled, only key-signing keys (i.e., keys with the KSK bit set) will be used to sign the DNSKEY RRset at the zone apex. Zone-signing keys (i.e., keys without the KSK bit set) will be used to sign the remainder of the zone, but not the DNSKEY RRset. This is similar to the **dnssec-signzone** program's `-x` command-line option. The default value is `no`. If `update-check-ksk`'s value is set to `no`, this option is ignored.

dnssec-loadkeys-interval <integer>;

When a zone has `auto-dnssec` configured to the value `maintain`, its key repository must be checked periodically to see if any new keys have been added or any existing keys' timing metadata has been updated (see *dnssec-keygen(1)* and *dnssec-settime(1)*). This option sets the frequency of automatic repository checks in minutes. The default value is 60. Values should be configured between 1 and 1440 inclusive; values greater than this range will be silently lowered to the maximum value.

dnssec-secure-to-insecure <boolean>;

Allow a dynamic zone to transition from secure to insecure (i.e., signed to unsigned) by deleting all of the DNSKEY records. The default value is `no`. If enabled, and if the DNSKEY RRset at the zone apex is deleted, all RRSIG and NSEC records will be removed from the zone as well.

Note

If the zone uses NSEC3, then it is also necessary to delete the NSEC3PARAM RRset from the zone apex; this will cause the removal of all corresponding NSEC3 records. (It is expected that this requirement will be eliminated in a future release.)

Error

TODO: Remove the need for the previous note.

Note

If a zone has `auto-dnssec` configured to `maintain` and the private keys remain accessible in the key repository, then the zone will be automatically signed again the next time **named** is started.

dnssec-update-mode (maintain | no-resign);

If this option is configured to `maintain` in a zone of type `master` which is DNSSEC-signed and configured to allow DNS UPDATES, and if **named** has access to the private signing key(s) for the zone, then it will automatically sign all new or changed records and maintain signatures for the zone by regenerating RRSIG records whenever they approach their expiration date.

If the option is configured to `no-resign`, then **named** will sign all new or changed records, but scheduled maintenance of signatures is disabled.

With either of these values, **named** will reject updates to a DNSSEC-signed zone when the signing keys are inactive or unavailable to it.

The default value is `maintain`.

Note

A planned third configuration value, `external`, will disable all automatic signing and allow DNSSEC data to be submitted into a zone via dynamic update; this is not yet implemented.

Error

TODO: Implement the `external` value?

forward (first | only);

This option is only used by **named** if the `forwarders` option's value is not empty. A value of `first` causes the resolver to query the forwarders first, and if that doesn't answer the question, the resolver will then attempt to resolve the answer by itself. A value of `only` causes the resolver to only query the forwarders. The default value is `first`.

forwarders [port <port:integer>] [dscp <dscp:integer>] { (<ipv4_addr...>);

Specifies a list of IP addresses of upstream nameservers to forward queries to. Forwarding occurs only on those queries for which the server is not authoritative and does not have the answer in its cache. It is not configured by default (no forwarding occurs).

inline-signing <boolean>;

TBD.

key-directory <quoted_string>;

When performing DNS UPDATES of secure zones, the directory where the public and private DNSSEC key files should be found, if it is different from the current working directory.

Note

This option has no effect on the paths for files containing non-DNSSEC keys such as `rndc.key` and `session.key`.

max-journal-size <size_no_default>;

Sets the maximum size for each journal file. When the journal file approaches the specified size, some of the oldest transactions in the journal will be automatically removed. The largest permitted value is 2 gigabytes. The default value is `unlimited`, which also means 2 gigabytes. See the Loop User Manual for a description of journal files.

max-records <integer>;

Sets the maximum number of resource records permitted in a zone. The default value is 0 which means unlimited.

max-refresh-time <integer>;

This option controls the server's behavior on refreshing a zone (querying for SOA changes) or retrying failed transfers. Usually the SOA values for the zone are used, up to a hard-coded maximum expiry of 24 weeks. However, these values are set by the master, giving slave server administrators little control over their contents.

This option allows an administrator to configure the maximum refresh time in seconds. It is valid for slave and stub zones, and clamps the SOA refresh time to the specified value. The default value is 2419200 (4 weeks).

max-retry-time <integer>;

This option controls the server's behavior on refreshing a zone (querying for SOA changes) or retrying failed transfers. Usually the SOA values for the zone are used, up to a hard-coded maximum expiry of 24 weeks. However, these values are set by the master, giving slave server administrators little control over their contents.

This option allows an administrator to configure the maximum retry time in seconds. It is valid for slave and stub zones, and clamps the SOA retry time to the specified value. The default value is 1209600 (2 weeks).

max-transfer-idle-in <integer>;

Sets the maximum idle time in minutes for inbound zone transfers, after which transfers making no progress will be terminated. The default value is 60. The maximum value is 40320 (28 days).

max-transfer-idle-out <integer>;

Sets the maximum idle time in minutes for outbound zone transfers, after which transfers making no progress will be terminated. The default value is 60. The maximum value is 40320 (28 days).

max-transfer-time-in <integer>;

Sets the maximum transfer time in minutes for inbound zone transfers, after

which transfers running longer will be terminated. The default value is 120. The maximum value is 40320 (28 days).

max-transfer-time-out <integer>;

Sets the maximum transfer time in minutes for outbound zone transfers, after which transfers running longer will be terminated. The default value is 120. The maximum value is 40320 (28 days).

max-zone-ttl (unlimited | <ttlval>);

Specifies a maximum permissible TTL value in seconds within a zone. When loading a zone file, any record encountered with a TTL higher than the configured value will cause the zone to be rejected.

This is useful in DNSSEC-signed zones because when rolling to a new DNSKEY, the old key needs to remain available until RRSIG records have expired from caches. This config option guarantees that the largest TTL in the zone will be no higher the set value.

The default value is *unlimited*. A value of 0 is treated as *unlimited*.

min-refresh-time <integer>;

This option controls the server's behavior on refreshing a zone (querying for SOA changes) or retrying failed transfers. Usually the SOA values for the zone are used, up to a hard-coded maximum expiry of 24 weeks. However, these values are set by the master, giving slave server administrators little control over their contents.

This option allows an administrator to configure the minimum refresh time in seconds. It is valid for slave and stub zones, and clamps the SOA refresh time to the specified value. The default value is 300.

min-retry-time <integer>;

This option controls the server's behavior on refreshing a zone (querying for SOA changes) or retrying failed transfers. Usually the SOA values for the zone are used, up to a hard-coded maximum expiry of 24 weeks. However, these values are set by the master, giving slave server administrators little control over their contents.

This option allows an administrator to configure the minimum retry time in seconds. It is valid for slave and stub zones, and clamps the SOA retry time to the specified value. The default value is 500.

multi-master <boolean>;

This option is meant to be used when there are multiple masters for a zone and the addresses refer to different machines. If enabled, **named** will not log when the serial number on the master is less than what **named** currently has. The default value is *no*.

notify (explicit | master-only | <boolean>);

This option configures generation of DNS NOTIFY messages:

- If configured as *yes*, DNS NOTIFY messages are sent when a zone that the nameserver is authoritative for changes. These messages are sent to the

nameservers listed in the zone's NS records (except the master server identified in the SOA MNAME field), and to any other nameservers configured using the `also-notify` option.

- If configured as `master-only`, DNS NOTIFY messages are only sent for master zones.
- If configured as `explicit`, notifies are sent only to servers explicitly listed using `also-notify`.
- If configured as `no`, no notifies are sent.

The default value is `yes`. See the Loop User Manual for a description of the DNS NOTIFY feature.

notify-delay <integer>;

Configures the delay in seconds between sending sets of DNS NOTIFY messages for a zone. The default value is 5.

The overall rate at which DNS NOTIFY messages are sent for all zones is controlled by the `serial-query-rate` option.

notify-source (<ipv4_address> | *) [port (<integer> | *)] [dscp <integer>]

Determines which local source IPv4 address, and optionally UDP port, are used to send DNS NOTIFY messages over IPv4. This IP address must appear in the slave nameserver's `masters` zone clause or in an `allow-notify` clause. It is not configured by default.

notify-source-v6 (<ipv6_address> | *) [port (<integer> | *)] [dscp <integer>]

Determines which local source IPv6 address, and optionally UDP port, are used to send DNS NOTIFY messages over IPv6. This IP address must appear in the slave nameserver's `masters` zone clause or in an `allow-notify` clause. It is not configured by default.

notify-to-soa <boolean>;

If enabled, **named** does not check the nameservers in the NS RRset against the SOA MNAME. Normally a DNS NOTIFY message is not sent to the nameserver in the SOA MNAME field as it is supposed to contain the name of the ultimate master. Sometimes, however, a slave nameserver is listed as the SOA MNAME in hidden master configurations and in that case you may want the ultimate master to still send NOTIFY messages to all the nameservers listed in the NS RRset.

nsec3-test-zone <boolean>; // test only

TBD.

Warning

This option is present for testing only. It is only of interest to Loop developers and may be removed or changed in a future release. Please don't use it.

request-ixfr <boolean>;

Whether the local nameserver, acting as a slave, will request IXFRs (incremental zone transfers; see [RFC 1995](#)) from the given remote nameserver, a master. The default value is *yes*.

IXFR requests to servers that do not support IXFR will automatically fall back to AXFR (non-incremental full zone transfer; see [RFC 5936](#)). Therefore, there is no need to explicitly list which servers support IXFR and which ones do not; the default of *yes* should always work. The purpose of the *provide-ixfr* and *request-ixfr* clauses is to make it possible to disable the use of IXFR even when both master and slave claim to support it, for example if one of the servers is buggy and crashes or corrupts data when IXFR is used.

serial-update-method (*increment* | *unixtime*);

Zones configured for DNS UPDATES may use this option to set the method that will be used to update the zone's SOA serial number.

- If configured as *increment*, the SOA serial number will be incremented by 1 each time the zone is updated.
- If configured as *unixtime*, the SOA serial number will be set to the number of seconds since the UNIX epoch, unless the serial number is already greater than or equal to that value, in which case it is incremented by 1.

The default value is *increment*.

Error

TODO: What about the YYYYMMDDnn format in [RFC 1912](#)?

sig-signing-nodes <integer>;

Specifies the maximum number of nodes to be examined in each quantum when signing a zone with a new DNSKEY. The default value is 100.

sig-signing-signatures <integer>;

Specifies a threshold number of signatures that will terminate processing a quantum when signing a zone with a new DNSKEY. The default value is 10.

sig-signing-type <integer>;

Specifies a RDATA type to be used when generating signing state records. Valid values are between 65280 and 65534 inclusive. The default value is 65534.

Note

It is expected that this parameter may be removed in a future version once there is a standard type.

Error

TODO: The range of values allowed is not enforced to be within the private use space.

Signing state records are internally used by **named** to track the current state of a zone-signing process, i.e., whether it is still active or has been completed.

- The records can be inspected using the **rndc** `signing -list` command.
- Once **named** has finished signing a zone with a particular key, the signing state record associated with that key can be removed from the zone using the **rndc** `signing -clear` command.
- To clear all of the completed signing state records for a zone, use the **rndc** `signing -clear all` command.

sig-validity-interval <validity:integer> [<integer>];

Specifies the number of days into the future when DNSSEC signatures automatically generated as a result of DNS UPDATES will expire, i.e., the signature validity interval.

There is an optional second field which specifies how long before expiry that the signatures will be regenerated. The second field is specified in days if *validity* is greater than 7 days otherwise it is specified in hours. If not specified, the signatures will be regenerated at 1/4 of *validity*.

The default value of *validity* (the signature validity interval) is 30, causing signatures to be re-regenerated every 7 1/2 days. The maximum value allowed for *validity* is 3660 (~10 years).

The signature inception time is unconditionally set to one hour before the current time to allow for a limited amount of clock skew.

The *validity* should be, at least, several multiples of the SOA expire interval to allow for reasonable interaction between the various timer and expiry dates.

transfer-source (<ipv4_address> | *) [port (<integer> | *)] [dscp

Determines which local IPv4 address will be bound to IPv4 TCP connections used to fetch zones transferred inbound by the server. It also determines the source IPv4 address, and optionally the UDP port, used for the SOA refresh queries and forwarded DNS UPDATES. If not set, it defaults to a system controlled value which will usually be the address of the interface "closest to" the remote end. It is not configured by default.

Note

This address should also be allowed in the remote nameserver's `allow-transfer` option for the zone being transferred.

transfer-source-v6 (<ipv6_address> | *) [port (<integer> | *)] [ds

Determines which local IPv6 address will be bound to IPv6 TCP connections used to fetch zones transferred inbound by the server. It also determines the source IPv6 address, and optionally the UDP port, used for the SOA refresh queries and forwarded DNS UPDATES. If not set, it defaults to a system controlled value which will usually be the address of the interface "closest to" the remote end. It is not configured by default.

Note

This address should also be allowed in the remote nameserver's `allow-transfer` option for the zone being transferred.

try-tcp-refresh <boolean>;

If enabled, **named** tries to send SOA refresh queries for zones using TCP, if SOA refresh queries over UDP fail. The default value is `yes`.

update-check-ksk <boolean>;

If enabled, **named** checks the KSK bit in each key to determine how the key should be used when generating RRSIGs for a secure zone. The default value is `yes`.

Ordinarily, zone-signing keys (i.e., keys without the KSK bit set) are used to sign the entire zone, while key-signing keys (i.e., keys with the KSK bit set) are only used to sign the DNSKEY RRset at the zone apex. However, if this option is set to `no`, then the KSK bit is ignored; KSKs are treated as if they were ZSKs and are used to sign the entire zone. This is similar to running **dnssec-signzone** with the `-z` argument.

When this option is set to `yes`, there must be at least two active keys for every algorithm represented in the DNSKEY RRset, i.e., at least one KSK and one ZSK per algorithm. If there is any algorithm for which this requirement is not met, this option will be ignored for that algorithm.

use-alt-transfer-source <boolean>;

Whether to use the `alt-transfer-source` and `alt-transfer-source-v6` options. If views are specified, the default value is `no`, otherwise the default value is `yes`.

zero-no-soa-ttl <boolean>;

If enabled, when returning authoritative negative responses to SOA queries, set the TTL of the SOA record returned in the authority section to 0. The default value is `yes`.

zone-statistics (full | terse | none | <boolean>);

This option configures what statistics data is collected for zones:

- If configured as `full` or `yes`, the nameserver will collect full zone statistics.

Error

TODO: This has to be described in more detail.

- If configured as `terse`, the nameserver will collect minimal statistics on zones (including name and current serial number, but not query type counters).
- If configured as `none` or `no`, the nameserver will not collect any statistics for zones.

Zone statistics may be accessed by running the `rndc stats` command, which will dump them to the file listed in the `statistics-file`. See the Loop User Manual for a description of the statistics data.

The default value is `terse`.

5.1.2.10 server statement

```
server <netprefix> {
    bogus <boolean>;
    edns <boolean>;
    edns-udp-size <integer>;
    keys <server_key>;
    max-udp-size <integer>;
    notify-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
    provide-ixfr <boolean>;
    query-source ( ( [ address ] ( <ipv4_address> | * ) [
↳port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_
↳address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>
↳ ];
    query-source-v6 ( ( [ address ] ( <ipv6_address> | *
↳ ) [ port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_
↳address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>
↳ ];
    request-ixfr <boolean>;
    request-nsid <boolean>;
    request-cookie <boolean>;
    tcp-only <boolean>;
    transfer-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
    transfers <integer>;
}; // may occur multiple times
```

The `server` statement defines characteristics to be associated with a remote name-server address. If a prefix length is specified as part of the address, then a range of servers is covered. The most specific `server` statement with the longest matching address prefix is used, regardless of the order in which the `server` statements appear in the config file.

The `server` statement can occur at the top-level of the configuration file or within a `view` statement. If a `view` statement contains one or more `server` statements, only those apply to the view and any top-level ones are ignored. If a view contains no `server` statements, any top-level `server` statements are used.

bogus <boolean>

If you discover that a remote server is giving out bad data, marking it as bogus will prevent further queries to it. The default value of `bogus` is `no`.

edns <boolean>

The `edns` clause determines whether the local server will attempt to use EDNS (see [RFC 6891](#)) when communicating with the remote server. The default value of `edns` is `yes`.

edns-udp-size <integer>

The `edns-udp-size` option sets the advertised EDNS requestor's UDP payload size in octets (see [RFC 6891](#) section 6.2.3) that is advertised by **named** when querying the remote nameserver. Valid values are between 512 and 4096 inclusive; values outside this range will be silently adjusted to the nearest value within it. This option is useful if you wish to advertise a different value to this server than the value you advertise globally, for example, when there is a firewall at the remote site that is blocking large replies.

Note

Currently, this sets a fixed UDP size for all packets sent to the server. **named** will not deviate from this value. This differs from the behavior of `edns-udp-size` in the `options` or `view` statements, where it specifies the maximum value for the advertised EDNS requestor's UDP payload size. The `server` statement behavior may be brought into conformance with the `options` and `view` statements' behavior in future releases.

keys <server_key>

The `keys` clause identifies a key defined by the `key` statement to be used to TSIG-sign DNS messages when talking to the remote nameserver. When a request is sent to the remote nameserver, a TSIG signature will be generated for it using the key specified here and appended to the message. A request originating from the remote nameserver is not required to be signed by this key.

Note

Only a single key per server is currently supported.

max-udp-size <integer>

The `max-udp-size` option sets the maximum EDNS UDP reply message size in octets that **named** will send to the remote nameserver. Valid values are between 512 and 4096 inclusive; values outside this range will be silently adjusted. This option is useful when you know that there is a firewall that is blocking large replies from **named** from getting through to the remote nameserver.

notify-source (<ipv4_address> | *) [port (<integer> | *)] [dscp <integer>]

The `notify-source` clause specifies the IPv4 source address to be used for NOTIFY messages sent to the remote nameserver.

Note

For an IPv6 remote server, this option must not be specified.

notify-source-v6 (<ipv6_address> | *) [port (<integer> | *)] [dscp <integer>]

The `notify-source-v6` clause specifies the IPv6 source address to be used for NOTIFY messages sent to the remote nameserver.

Note

For an IPv4 remote server, this option must not be specified.

provide-ixfr <boolean>

The `provide-ixfr` clause determines whether the local nameserver, when acting as master, will respond with IXFRs (incremental zone transfers; see [RFC 1995](#)) when the given remote nameserver, a slave or client, requests it. If set to `yes`, incremental transfer will be provided whenever possible. If set to `no`, all transfers to the remote server or client will be AXFRs (non-incremental full zone transfer; see [RFC 5936](#)). If not set, the value of the `provide-ixfr` option in the `view` or `options` statement is used as the default.

Error

TODO: It may also be set in the `zone` statement and, if set there, it will override the `options` or `view` statement's setting for that zone?

query-source (([address] (<ipv4_address> | *) [port (<integer> | *)]))

The `query-source` clause specifies the IPv4 source address to be used for DNS queries sent to the remote nameserver.

Note

For an IPv6 remote server, this option must not be specified.

query-source-v6 (([address] (<ipv6_address> | *) [port (<integer>

The `query-source-v6` clause specifies the IPv6 source address to be used for DNS queries sent to the remote nameserver.

Note

For an IPv4 remote server, this option must not be specified.

request-ixfr <boolean>

Overrides the `request-ixfr` value set at the view or options statement level. See the documentation for the `request-ixfr` option in the options statement.

request-nsid <boolean>

Overrides the `request-nsid` value set at the view or options statement level. See the documentation for the `request-nsid` option in the options statement.

request-cookie <boolean>

Overrides the `request-cookie` value set at the view or options statement level. See the documentation for the `request-cookie` option in the options statement.

tcp-only <boolean>

The `tcp-only` option forces the transport protocol to TCP. The default is to use the UDP transport and to fallback on TCP only if a truncated response is received.

transfer-source (<ipv4_address> | *) [port (<integer> | *)] [dscp

The `transfer-source` clause specifies the IPv4 source address to be used for zone transfers with the remote nameserver.

Note

For an IPv6 remote server, this option must not be specified.

transfer-source-v6 (<ipv6_address> | *) [port (<integer> | *)] [ds

The `transfer-source-v6` clause specifies the IPv6 source address to be used for zone transfers with the remote nameserver.

Note

For an IPv4 remote server, this option must not be specified.

transfers <integer>

The `transfers` option is used to limit the number of concurrent inbound zone transfers from the remote nameserver. If no `transfers` clause is specified, the limit is set according to the `transfers-per-ns` option.

5.1.2.11 trusted-keys statement

```
trusted-keys { <name:string> <flags:integer>
↳<protocol:integer> <algorithm:integer> <key:quoted_string>;
↳... }; // may occur multiple times
```

The `trusted-keys` statement configures static DNSSEC trust anchors. A trust anchor may be defined when the DNSKEY for a zone is known, but cannot be securely obtained through DNS, either because it is the DNS root zone or because its parent zone is unsigned. Once a key has been configured as a trusted-key, it is treated as if it had been validated and proven secure. The resolver attempts DNSSEC validation on all DNS data in sub-domains of a trust anchor's domain.

All keys (and corresponding zones) listed in `trusted-keys` are deemed to exist regardless of what parent zones say. Similarly for all keys listed in `trusted-keys`, only those keys are used to validate the DNSKEY RRset. The parent's DS RRset will not be used.

The `trusted-keys` statement may contain multiple keys, each consisting of the key's domain name, flags, protocol, algorithm, and the Base64 representation of the key data. Spaces, tabs, newlines and carriage returns are ignored in the key data, so the configuration may be split up into multiple lines.

`trusted-keys` may be set at the top level of the config file or within a `view` statement. If it is set in both places, they are additive, i.e., keys defined at the top-level are inherited by all `view` statements, but keys defined in a `view` statement are only used within that view.

5.1.2.12 view statement

```
view <name:string> [ <class> ] {
    match-clients { <address_match_element>; ... };
    match-destinations { <address_match_element>; ... };
    match-recursive-only <boolean>;
    key <string> {
        algorithm <string>;
        secret <string>;
    }; // may occur multiple times
    managed-keys { <name:string> <init:string>
↳<flags:integer> <protocol:integer> <algorithm:integer>
↳<key:quoted_string>; ... }; // may occur multiple times
    server <netprefix> {
        bogus <boolean>;
        edns <boolean>;
```

(continues on next page)

(continued from previous page)

```

        edns-udp-size <integer>;
        keys <server_key>;
        max-udp-size <integer>;
        notify-source ( <ipv4_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
        notify-source-v6 ( <ipv6_address> | * ) [
↪port ( <integer> | * ) ] [ dscp <integer> ];
        provide-ixfr <boolean>;
        query-source ( ( [ address ] ( <ipv4_address>
↪| * ) [ port ( <integer> | * ) ] ) | ( [ [ address ] (
↪<ipv4_address> | * ) ] port ( <integer> | * ) ) ) [ dscp
↪<integer> ];
        query-source-v6 ( ( [ address ] ( <ipv6_
↪address> | * ) [ port ( <integer> | * ) ] ) | ( [ [ address
↪] ( <ipv6_address> | * ) ] port ( <integer> | * ) ) ) [
↪dscp <integer> ];
        request-ixfr <boolean>;
        request-nsid <boolean>;
        request-cookie <boolean>;
        tcp-only <boolean>;
        transfer-source ( <ipv4_address> | * ) [ port
↪( <integer> | * ) ] [ dscp <integer> ];
        transfer-source-v6 ( <ipv6_address> | * ) [
↪port ( <integer> | * ) ] [ dscp <integer> ];
        transfers <integer>;
    }; // may occur multiple times
    trusted-keys { <name:string> <flags:integer>
↪<protocol:integer> <algorithm:integer> <key:quoted_string>;
↪... }; // may occur multiple times
    zone <name:string> [ <class> ] {
        type ( delegation-only | forward | hint |
↪master | slave | static-stub | stub );
        file <quoted_string>;
        journal <quoted_string>;
        masters [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↪<integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↪<key:string> ]; ... };
        update-policy ( local | { ( deny | grant )
↪<identity:string> ( 6to4-self | external | krb5-self | krb5-
↪subdomain | ms-self | ms-subdomain | name | self | selfsub
↪| selfwild | subdomain | tcp-self | wildcard | zonesub ) [
↪<name:string> ] <types:rrtypelist>; ... };
        database <string>;
        delegation-only <boolean>;
        check-names ( fail | warn | ignore );

```

(continues on next page)

(continued from previous page)

```

        in-view <string>;
        ixfr-from-differences <boolean>;
        server-addresses { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ]; ... };
        server-names { <quoted_string>; ... };
        allow-notify { <address_match_element>; ... };
        allow-query { <address_match_element>; ... };
        allow-query-on { <address_match_element>; ... ↪
↪};
        allow-transfer { <address_match_element>; ... ↪
↪};
        allow-update { <address_match_element>; ... };
        allow-update-forwarding { <address_match_
↪element>; ... };
        also-notify [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↪<integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↪<key:string> ]; ... };
        alt-transfer-source ( <ipv4_address> | * ) [ ↪
↪port ( <integer> | * ) ] [ dscp <integer> ];
        alt-transfer-source-v6 ( <ipv6_address> | * ) ↪
↪[ port ( <integer> | * ) ] [ dscp <integer> ];
        auto-dnssec ( allow | maintain | off );
        check-dup-records ( fail | warn | ignore );
        check-integrity <boolean>;
        check-mx ( fail | warn | ignore );
        check-mx-cname ( fail | warn | ignore );
        check-sibling <boolean>;
        check-spf ( warn | ignore );
        check-srv-cname ( fail | warn | ignore );
        check-wildcard <boolean>;
        dialup ( notify | notify-passive | passive | ↪
↪refresh | <boolean> );
        dnssec-dnskey-kskonly <boolean>;
        dnssec-loadkeys-interval <integer>;
        dnssec-secure-to-insecure <boolean>;
        dnssec-update-mode ( maintain | no-resign );
        forward ( first | only );
        forwarders [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <ipv4_address> | <ipv6_address> ) [ ↪
↪port <integer> ] [ dscp <integer> ]; ... };
        inline-signing <boolean>;
        key-directory <quoted_string>;
        max-journal-size <size_no_default>;
        max-records <integer>;
        max-refresh-time <integer>;

```

(continues on next page)

(continued from previous page)

```

max-retry-time <integer>;
max-transfer-idle-in <integer>;
max-transfer-idle-out <integer>;
max-transfer-time-in <integer>;
max-transfer-time-out <integer>;
max-zone-ttl ( unlimited | <ttlval> );
min-refresh-time <integer>;
min-retry-time <integer>;
multi-master <boolean>;
notify ( explicit | master-only | <boolean> );
notify-delay <integer>;
notify-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [
↳port ( <integer> | * ) ] [ dscp <integer> ];
    notify-to-soa <boolean>;
nsec3-test-zone <boolean>; // test only
request-ixfr <boolean>;
serial-update-method ( increment | unixtime );
sig-signing-nodes <integer>;
sig-signing-signatures <integer>;
sig-signing-type <integer>;
sig-validity-interval <validity:integer> [
↳<integer> ];
    transfer-source ( <ipv4_address> | * ) [ port
↳( <integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [
↳port ( <integer> | * ) ] [ dscp <integer> ];
    try-tcp-refresh <boolean>;
    update-check-ksk <boolean>;
    use-alt-transfer-source <boolean>;
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none |
↳<boolean> );
    }; // may occur multiple times
allow-new-zones <boolean>;
allow-query-cache { <address_match_element>; ... };
allow-query-cache-on { <address_match_element>; ... };
allow-recursion { <address_match_element>; ... };
allow-recursion-on { <address_match_element>; ... };
attach-cache <string>;
auth-nxdomain <boolean>; // default changed
cache-file <quoted_string>; // test only
check-names ( master | slave | response ) ( fail |
↳warn | ignore ); // may occur multiple times
clients-per-query <integer>;

```

(continues on next page)

(continued from previous page)

```

    deny-answer-addresses { <acl:address_match_element>; .
↪... } [ except-from { <except-from:quoted_string>; ... } ];
    deny-answer-aliases { <name:quoted_string>; ... } [
↪except-from { <except-from:quoted_string>; ... } ];
    disable-algorithms <domain:string> {
↪<algorithms:string>; ... }; // may occur multiple times
    disable-ds-digests <domain:string> { <digests:string>;
↪... }; // may occur multiple times
    disable-empty-zone <string>; // may occur multiple
↪times
    dns64 <netprefix> {
        break-dnssec <boolean>;
        clients { <address_match_element>; ... };
        exclude { <address_match_element>; ... };
        mapped { <address_match_element>; ... };
        recursive-only <boolean>;
        suffix <ipv6_address>;
    }; // may occur multiple times
    dns64-contact <string>;
    dns64-server <string>;
    dnssec-enable <boolean>;
    dnssec-must-be-secure <domain:string> <value:boolean>;
↪ // may occur multiple times
    dnssec-validation ( yes | no | auto );
    dual-stack-servers [ port <port:integer> ] { (
↪<quoted_string> [ port <integer> ] [ dscp <integer> ] |
↪<ipv4_address> [ port <integer> ] [ dscp <integer> ] |
↪<ipv6_address> [ port <integer> ] [ dscp <integer> ] ); ...
↪};

    edns-udp-size <integer>;
    empty-contact <string>;
    empty-server <string>;
    empty-zones-enable <boolean>;
    fetch-quota-params <frequency:integer>
↪<low:fixedpoint> <high:fixedpoint> <discount:fixedpoint>;
    fetches-per-server <fetches:integer> [ ( drop | fail
↪) ];
    fetches-per-zone <fetches:integer> [ ( drop | fail )
↪];

    ixfr-from-differences ( master | slave | <boolean> );
    max-cache-size <size_no_default>;
    max-cache-ttl <integer>;
    max-clients-per-query <integer>;
    max-ncache-ttl <integer>;
    max-recursion-depth <integer>;
    max-recursion-queries <integer>;

```

(continues on next page)

(continued from previous page)

```

max-udp-size <integer>;
minimal-responses <boolean>;
no-case-compress { <address_match_element>; ... };
nocookie-udp-size <integer>;
preferred-glue <string>;
prefetch <trigger:integer> [ <integer> ];
provide-ixfr <boolean>;
query-source ( ( [ address ] ( <ipv4_address> | * ) [ _
↪port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_
↪address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>
↪ ];
query-source-v6 ( ( [ address ] ( <ipv6_address> | * _
↪) [ port ( <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_
↪address> | * ) ] port ( <integer> | * ) ) ) [ dscp <integer>
↪ ];
rate-limit {
    all-per-second <integer>;
    errors-per-second <integer>;
    exempt-clients { <address_match_element>; ... _
↪};

    ipv4-prefix-length <integer>;
    ipv6-prefix-length <integer>;
    log-only <boolean>;
    max-table-size <integer>;
    min-table-size <integer>;
    nodata-per-second <integer>;
    nxdomains-per-second <integer>;
    qps-scale <integer>;
    referrals-per-second <integer>;
    responses-per-second <integer>;
    slip <integer>;
    window <integer>;
};
recursion <boolean>;
request-nsid <boolean>;
request-cookie <boolean>;
resolver-query-timeout <integer>;
response-policy { zone <quoted_string> [ policy ( _
↪cname | disabled | drop | given | no-op | nodata | nxdomain_
↪| passthru | tcp-only <cname:quoted_string> ) ] [ recursive-
↪only <boolean> ] [ max-policy-ttl <integer> ]; ... } [ _
↪recursive-only <boolean> ] [ break-dnssec <boolean> ] [ max-
↪policy-ttl <integer> ] [ min-ns-dots <integer> ] [ qname-
↪wait-recurse <boolean> ];
    root-delegation-only [ exclude { <quoted_string>; ... _
↪} ];

```

(continues on next page)

(continued from previous page)

```

rrset-order ( none | random );
zero-no-soa-ttl-cache <boolean>;
allow-notify { <address_match_element>; ... };
allow-query { <address_match_element>; ... };
allow-query-on { <address_match_element>; ... };
allow-transfer { <address_match_element>; ... };
allow-update { <address_match_element>; ... };
allow-update-forwarding { <address_match_element>; ...
↪ };
    also-notify [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↪<integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↪<key:string> ]; ... };
        alt-transfer-source ( <ipv4_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
        alt-transfer-source-v6 ( <ipv6_address> | * ) [ port
↪( <integer> | * ) ] [ dscp <integer> ];
        auto-dnssec ( allow | maintain | off );
        check-dup-records ( fail | warn | ignore );
        check-integrity <boolean>;
        check-mx ( fail | warn | ignore );
        check-mx-cname ( fail | warn | ignore );
        check-sibling <boolean>;
        check-spf ( warn | ignore );
        check-srv-cname ( fail | warn | ignore );
        check-wildcard <boolean>;
        dialup ( notify | notify-passive | passive | refresh
↪| <boolean> );
        dnssec-dnskey-kskonly <boolean>;
        dnssec-loadkeys-interval <integer>;
        dnssec-secure-to-insecure <boolean>;
        dnssec-update-mode ( maintain | no-resign );
        forward ( first | only );
        forwarders [ port <port:integer> ] [ dscp
↪<dscp:integer> ] { ( <ipv4_address> | <ipv6_address> ) [
↪port <integer> ] [ dscp <integer> ]; ... };
        inline-signing <boolean>;
        key-directory <quoted_string>;
        max-journal-size <size_no_default>;
        max-records <integer>;
        max-refresh-time <integer>;
        max-retry-time <integer>;
        max-transfer-idle-in <integer>;
        max-transfer-idle-out <integer>;
        max-transfer-time-in <integer>;
        max-transfer-time-out <integer>;

```

(continues on next page)

(continued from previous page)

```
max-zone-ttl ( unlimited | <ttlval> );
min-refresh-time <integer>;
min-retry-time <integer>;
multi-master <boolean>;
notify ( explicit | master-only | <boolean> );
notify-delay <integer>;
notify-source ( <ipv4_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
notify-to-soa <boolean>;
nsec3-test-zone <boolean>; // test only
request-ixfr <boolean>;
serial-update-method ( increment | unixtime );
sig-signing-nodes <integer>;
sig-signing-signatures <integer>;
sig-signing-type <integer>;
sig-validity-interval <validity:integer> [ <integer>↪
↪];
    transfer-source ( <ipv4_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port (
↪<integer> | * ) ] [ dscp <integer> ];
    try-tcp-refresh <boolean>;
    update-check-ksk <boolean>;
    use-alt-transfer-source <boolean>;
    zero-no-soa-ttl <boolean>;
    zone-statistics ( full | terse | none | <boolean> );
}; // may occur multiple times
```

The `view` statement is a powerful feature of Loop that lets a nameserver answer a DNS query differently depending on who is asking. It is particularly useful for implementing split-DNS setups without having to run multiple servers.

Each `view` statement defines a view of the DNS namespace that will be seen by a subset of the nameserver's clients. A client matches a view if its source IP address matches the view's `match-clients` option value, and its destination IP address matches the view's `match-destinations` option value. In addition to checking IP addresses `match-clients` and `match-destinations` can also take keys which provide a mechanism to select the view for a query based on the TSIG key with which it was signed. A view can also be specified as `match-recursive-only`, which means that only recursive requests from matching clients will match that view.

The lexical order of the `view` statements is significant. A client query will be resolved in the context of the first `view` that it matches in the configuration file.

Zones defined within a `view` statement will only be accessible to clients that match the view. By defining a zone of the same name in multiple views, different zone data

can be returned to different clients, for example, "internal" and "external" clients in a split-DNS setup.

Many of the options given in the `options` statement can also be used within a `view` statement, and they then apply only when processing queries with that view. When no view-specific value is given, the value in the `options` statement is used as a default. Also, `zone` options can have default values specified in the `view` statement; values specified at the `view` statement level take precedence over those in the `options` statement.

Views are class specific. If no class is given, class IN is assumed. All non-IN views must contain a hint zone, since only the IN class has compiled-in default hints.

Error

TODO: Is the hint zone necessary when recursion is not enabled?

If there are no explicit `view` statements in the config file, an implicit default view with the name `_default` that matches any client is automatically created in class IN. Any `zone` statements specified at the top-level of the configuration file are considered to be part of this default view, and the `options` statement's config values will apply to the default view.

Note

If any explicit `view` statements are present, all `zone` statements must occur inside `view` statements.

Error

TODO: Document the view-specific options as a list below.

5.1.2.13 zone statement

```
zone <name:string> [ <class> ] {
    type ( delegation-only | forward | hint | master |
↳slave | static-stub | stub );
    file <quoted_string>;
    journal <quoted_string>;
    masters [ port <port:integer> ] [ dscp <dscp:integer>
↳] { ( <masters> | <ipv4_address> [ port <integer> ] | <ipv6_
↳address> [ port <integer> ] ) [ key <key:string> ]; ... };
    update-policy ( local | { ( deny | grant )
↳<identity:string> ( 6to4-self | external | krb5-self | krb5-
↳subdomain | ms-self | ms-subdomain | name | self | selfsub
↳
```

(continues on next page)

(continued from previous page)

```

↪ | selfwild | subdomain | tcp-self | wildcard | zonesub ) [
↪ <name:string> ] <types:rrtypelist>; ... };
    database <string>;
    delegation-only <boolean>;
    check-names ( fail | warn | ignore );
    in-view <string>;
    ixfr-from-differences <boolean>;
    server-addresses { ( <ipv4_address> | <ipv6_address>
↪ ) [ port <integer> ]; ... };
    server-names { <quoted_string>; ... };
    allow-notify { <address_match_element>; ... };
    allow-query { <address_match_element>; ... };
    allow-query-on { <address_match_element>; ... };
    allow-transfer { <address_match_element>; ... };
    allow-update { <address_match_element>; ... };
    allow-update-forwarding { <address_match_element>; ...
↪ };
    also-notify [ port <port:integer> ] [ dscp
↪ <dscp:integer> ] { ( <masters> | <ipv4_address> [ port
↪ <integer> ] | <ipv6_address> [ port <integer> ] ) [ key
↪ <key:string> ]; ... };
    alt-transfer-source ( <ipv4_address> | * ) [ port (
↪ <integer> | * ) ] [ dscp <integer> ];
    alt-transfer-source-v6 ( <ipv6_address> | * ) [ port
↪ ( <integer> | * ) ] [ dscp <integer> ];
    auto-dnssec ( allow | maintain | off );
    check-dup-records ( fail | warn | ignore );
    check-integrity <boolean>;
    check-mx ( fail | warn | ignore );
    check-mx-cname ( fail | warn | ignore );
    check-sibling <boolean>;
    check-spf ( warn | ignore );
    check-srv-cname ( fail | warn | ignore );
    check-wildcard <boolean>;
    dialup ( notify | notify-passive | passive | refresh
↪ | <boolean> );
    dnssec-dnskey-kskonly <boolean>;
    dnssec-loadkeys-interval <integer>;
    dnssec-secure-to-insecure <boolean>;
    dnssec-update-mode ( maintain | no-resign );
    forward ( first | only );
    forwarders [ port <port:integer> ] [ dscp
↪ <dscp:integer> ] { ( <ipv4_address> | <ipv6_address> ) [
↪ port <integer> ] [ dscp <integer> ]; ... };
    inline-signing <boolean>;
    key-directory <quoted_string>;

```

(continues on next page)

(continued from previous page)

```

max-journal-size <size_no_default>;
max-records <integer>;
max-refresh-time <integer>;
max-retry-time <integer>;
max-transfer-idle-in <integer>;
max-transfer-idle-out <integer>;
max-transfer-time-in <integer>;
max-transfer-time-out <integer>;
max-zone-ttl ( unlimited | <ttlval> );
min-refresh-time <integer>;
min-retry-time <integer>;
multi-master <boolean>;
notify ( explicit | master-only | <boolean> );
notify-delay <integer>;
notify-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
  notify-source-v6 ( <ipv6_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
  notify-to-soa <boolean>;
  nsec3-test-zone <boolean>; // test only
  request-ixfr <boolean>;
  serial-update-method ( increment | unixtime );
  sig-signing-nodes <integer>;
  sig-signing-signatures <integer>;
  sig-signing-type <integer>;
  sig-validity-interval <validity:integer> [ <integer>
↳];
  transfer-source ( <ipv4_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
  transfer-source-v6 ( <ipv6_address> | * ) [ port (
↳<integer> | * ) ] [ dscp <integer> ];
  try-tcp-refresh <boolean>;
  update-check-ksk <boolean>;
  use-alt-transfer-source <boolean>;
  zero-no-soa-ttl <boolean>;
  zone-statistics ( full | terse | none | <boolean> );
}; // may occur multiple times

```

5.1.2.13.1 Zone Types

The type keyword is required for the zone configuration unless it is an in-view configuration. Its acceptable values include: delegation-only, forward, hint, master, slave, static-stub, and stub.

mas: The server has a master copy of the data for the zone and will be able to provide authoritative answers for it.

sla: A slave zone is a replica of a master zone. The `masters` list specifies one or more IP addresses of master servers that the slave contacts to update its copy of the zone. Masters list elements can also be names of other masters lists. By default, transfers are made from port 53 on the servers; this can be changed for all servers by specifying a port number before the list of IP addresses, or on a per-server basis after the IP address. Authentication to the master can also be done with per-server TSIG keys. If a file is specified, then the replica will be written to this file whenever the zone is changed, and reloaded from this file on a server restart. Use of a file is recommended, since it often speeds server startup and eliminates a needless waste of bandwidth. Note that for large numbers (in the tens or hundreds of thousands) of zones per server, it is best to use a two-level naming scheme for zone filenames. For example, a slave server for the zone `example.com` might place the zone contents into a file called `ex/example.com` where `ex/` is just the first two letters of the zone name. (Most operating systems behave very slowly if you put 100000 files into a single directory.)

stub: A stub zone is similar to a slave zone, except that it replicates only the NS records of a master zone instead of the entire zone. Stub zones are not a standard part of the DNS; they are a feature specific to the Loop implementation. Stub zones can be used to eliminate the need for glue NS record in a parent zone at the expense of maintaining a stub zone entry and a set of name server addresses in `named.conf`. This usage is not recommended for new configurations, and Loop supports it only in a limited way. If a Loop master serving a parent zone has child stub zones configured, all the slave servers for the parent zone also need to have the same child stub zones configured. Stub zones can also be used as a way of forcing the resolution of a given domain to use a particular set of authoritative servers. For example, the caching name servers on a private network using RFC1918 addressing may be configured with stub zones for `10.in-addr.arpa` to use a set of internal name servers as the authoritative servers for that domain.

sta: A static-stub zone is similar to a stub zone with the following exceptions: the zone data is statically configured, rather than transferred from a master server; when recursion is necessary for a query that matches a static-stub zone, the locally configured data (nameserver names and glue addresses) is always used even if different authoritative information is cached.

Zone data is configured via the `server-addresses` and `server-names` zone options.

The zone data is maintained in the form of NS and (if necessary) glue A or AAAA RRs internally, which can be seen by dumping zone databases by `rndc dumpdb -all`. The configured RRs are considered local configuration parameters rather than public data. Non recursive queries (i.e., those with the RD bit off) to a static-stub zone are therefore prohibited and will be responded with REFUSED.

Since the data is statically configured, no zone maintenance action takes place for a static-stub zone. For example, there is no periodic refresh attempt, and an incoming notify message will be rejected with an rcode of NOTAUTH.

Each static-stub zone is configured with internally generated NS and (if necessary) glue A or AAAA RRs

for: A "forward zone" is a way to configure forwarding on a per-domain basis. A zone statement of type `forward` can contain a `forward` and/or

5.1.2.13.2 Class

The zone's name may optionally be followed by a class. If a class is not specified, class IN (for Internet), is assumed. This is correct for the vast majority of cases.

The `hesiod` class is named for an information service from MIT's Project Athena. It is used to share information about various systems databases, such as users, groups, printers and so on. The keyword `HS` is a synonym for `hesiod`.

Another MIT development is Chaosnet, a LAN protocol created in the mid-1970s. Zone data for it can be specified with the `CHAOS` class.

5.1.2.13.3 Zone Options

allow-notify

See the description of `allow-notify` in *section_title*.

allow-query

See the description of `allow-query` in *section_title*.

allow-query-on

See the description of `allow-query-on` in *section_title*.

allow-transfer

See the description of `allow-transfer` in *section_title*.

allow-update

See the description of `allow-update` in *section_title*.

update-policy

Specifies a "Simple Secure Update" policy. See *section_title*.

allow-update-forwarding

See the description of `allow-update-forwarding` in *section_title*.

also-notify

Only meaningful if `notify` is active for this zone. The set of machines that will receive a DNS NOTIFY message for this zone is made up of all the listed name servers (other than the primary master) for the zone plus any IP addresses specified with `also-notify`. A port may be specified with each `also-notify` address to send the notify messages to a port other than the default of 53. A TSIG key may also be specified to cause the NOTIFY to be signed by the given key. `also-notify` is not meaningful for stub zones. The default is the empty list.

check-names

This option is used to restrict the character set and syntax of certain domain names in master files and/or DNS responses received from the network. The default varies according to zone type. For master zones the default is `fail`. For slave zones the default is `warn`. It is not implemented for hint zones.

check-mx

See the description of `check-mx` in *section_title*.

check-spf

See the description of `check-spf` in *section_title*.

check-wildcard

See the description of `check-wildcard` in *section_title*.

check-integrity

See the description of `check-integrity` in *section_title*.

check-sibling

See the description of `check-sibling` in *section_title*.

zero-no-soa-ttl

See the description of `zero-no-soa-ttl` in *section_title*.

update-check-ksk

See the description of `update-check-ksk` in *section_title*.

dnssec-loadkeys-interval

See the description of `dnssec-loadkeys-interval` in *section_title*.

dnssec-update-mode

See the description of `dnssec-update-mode` in *section_title*.

dnssec-dnskey-kskonly

See the description of `dnssec-dnskey-kskonly` in *section_title*.

try-tcp-refresh

See the description of `try-tcp-refresh` in *section_title*.

database

Specify the type of database to be used for storing the zone data. The string following the `database` keyword is interpreted as a list of whitespace-delimited words. The first word identifies the database type, and any subsequent words are passed as arguments to the database to be interpreted in a way specific to the database type.

The default is `"rbt"`, Loop's native in-memory red-black-tree database. This database does not take arguments.

Other values are possible if additional database drivers have been linked into the server. Some sample drivers are included with the distribution but none are linked in by default.

dialup

See the description of `dialup` in *section_title*.

delegation-only

The flag only applies to forward, hint and stub zones. If set to `yes`, then the zone will also be treated as if it is also a delegation-only type zone.

See caveats in *varlistentry_title*.

file

Set the zone's filename. In `master` and `hint` zones which do not have `masters` defined, zone data is loaded from this file. In `slave` and `stub` zones which do

have `masters` defined, zone data is retrieved from another server and saved in this file. This option is not applicable to other zone types.

forward

Only meaningful if the zone has a `forwarders` list. The `only` value causes the lookup to fail after trying the `forwarders` and getting no answer, while `first` would allow a normal lookup to be tried.

forwarders

Used to override the list of global forwarders. If it is not specified in a zone of type `forward`, no forwarding is done for the zone and the global options are not used.

journal

Allow the default journal's filename to be overridden. The default is the zone's filename with `".jnl"` appended. This is applicable to `master` and `slave` zones.

max-journal-size

See the description of `max-journal-size` in *section_title*.

max-records

See the description of `max-records` in *section_title*.

max-transfer-time-in

See the description of `max-transfer-time-in` in *section_title*.

max-transfer-idle-in

See the description of `max-transfer-idle-in` in *section_title*.

max-transfer-time-out

See the description of `max-transfer-time-out` in *section_title*.

max-transfer-idle-out

See the description of `max-transfer-idle-out` in *section_title*.

notify

See the description of `notify` in *section_title*.

notify-delay

See the description of `notify-delay` in *section_title*.

notify-to-soa

See the description of `notify-to-soa` in *section_title*.

zone-statistics

See the description of `zone-statistics` in *section_title*.

server-addresses

Only meaningful for static-stub zones. This is a list of IP addresses to which queries should be sent in recursive resolution for the zone. A non empty list for this option will internally configure the apex NS RR with associated glue A or AAAA RRs.

For example, if "example.com" is configured as a static-stub zone with 192.0.2.1 and 2001:db8::1234 in a `server-addresses` option, the following RRs will be

internally configured.

```
example.com. NS example.com.  
example.com. A 192.0.2.1  
example.com. AAAA 2001:db8::1234
```

These records are internally used to resolve names under the static-stub zone. For instance, if the server receives a query for "www.example.com" with the RD bit on, the server will initiate recursive resolution and send queries to 192.0.2.1 and/or 2001:db8::1234.

server-names

Only meaningful for static-stub zones. This is a list of domain names of name-servers that act as authoritative servers of the static-stub zone. These names will be resolved to IP addresses when `named` needs to send queries to these servers. To make this supplemental resolution successful, these names must not be a sub-domain of the origin name of static-stub zone. That is, when "example.net" is the origin of a static-stub zone, "ns.example" and "master.example.com" can be specified in the `server-names` option, but "ns.example.net" cannot, and will be rejected by the configuration parser.

A non empty list for this option will internally configure the apex NS RR with the specified names. For example, if "example.com" is configured as a static-stub zone with "ns1.example.net" and "ns2.example.net" in a `server-names` option, the following RRs will be internally configured.

```
example.com. NS ns1.example.net.  
example.com. NS ns2.example.net.
```

These records are internally used to resolve names under the static-stub zone. For instance, if the server receives a query for "www.example.com" with the RD bit on, the server initiate recursive resolution, resolve "ns1.example.net" and/or "ns2.example.net" to IP addresses, and then send queries to (one or more of) these addresses.

sig-validity-interval

See the description of `sig-validity-interval` in *section_title*.

sig-signing-nodes

See the description of `sig-signing-nodes` in *section_title*.

sig-signing-signatures

See the description of `sig-signing-signatures` in *section_title*.

sig-signing-type

See the description of `sig-signing-type` in *section_title*.

transfer-source

See the description of `transfer-source` in *section_title*.

transfer-source-v6

See the description of `transfer-source-v6` in *section_title*.

alt-transfer-source

See the description of alt-transfer-source in *section_title*.

alt-transfer-source-v6

See the description of alt-transfer-source-v6 in *section_title*.

use-alt-transfer-source

See the description of use-alt-transfer-source in *section_title*.

notify-source

See the description of notify-source in *section_title*.

notify-source-v6

See the description of notify-source-v6 in *section_title*.

min-refresh-time;max-refresh-time;min-retry-time;max-retry-time

See the description in *section_title*.

ixfr-from-differences

See the description of ixfr-from-differences in *section_title*. (Note that the ixfr-from-differences master and slave choices are not available at the zone level.)

key-directory

See the description of key-directory in *section_title*.

auto-dnssec

See the description of auto-dnssec in *section_title*.

serial-update-method

See the description of serial-update-method in *section_title*.

inline-signing

If yes, this enables "bump in the wire" signing of a zone, where a unsigned zone is transferred in or loaded from disk and a signed version of the zone is served, with possibly, a different serial number. This behaviour is disabled by default.

multi-master

See the description of multi-master in *section_title*.

max-zone-ttl

See the description of max-zone-ttl in *section_title*.

dnssec-secure-to-insecure

See the description of dnssec-secure-to-insecure in *section_title*.

5.1.3 Comments syntax

The comment syntax allows for comments to appear anywhere that whitespace may appear in a config file. To appeal to all programmers, they can be written in the C, C++, or shell/Perl style:

- C-style comments start with the two characters /* (slash, star) and end with */ (star, slash). Because they are completely delimited within these characters, they

can be used to comment only a portion of a line or to span multiple lines. For example:

```
/*  
 * This is a multi-line  
 * comment.  
 */
```

C-style comments cannot be nested. For example, the following syntax is not valid because the entire comment ends with the first `*/`:

```
/* This is the start of a comment.  
   This is still part of the comment.  
/* This is an incorrect attempt at nesting a comment. */  
   This is no longer in any comment. */
```

- C++-style comments start with the two characters `//` (slash, slash) and continue to the end of the physical line. They cannot be continued across multiple physical lines; to have one logical comment span multiple lines, each line must use the `//` pair. For example:

```
// This is a comment. It continues to end of line.  
// This next line is a new comment, even though it is  
// logically part of the previous comment.
```

- Shell-style or Perl-style comments start with the character `#` (number/hash sign) and continue to the end of the physical line. They cannot be continued across multiple physical lines; to have one logical comment span multiple lines, each line must use the `#` character. For example:

```
# This is a comment. It continues to end of line.  
# This next line is a new comment, even though it is  
# logically part of the previous comment.
```

Note

You cannot use the `;` (semi-colon) character to start a comment such as you would in a zone file. The semicolon indicates the end of a configuration statement.

5.1.4 Files

`/etc/loop/named.conf`

The configuration file for the `named(8)` program.

5.1.5 See also

ddns-confgen(1), *named(8)*, *named-checkconf(1)*, *rndc(1)*, *rndc.conf(5)*, *rndc-confgen(1)*

5.1.6 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2004-2018 Internet Systems Consortium, Inc. ("ISC").

5.2 *rndc.conf* --- *rndc* program's configuration

5.2.1 Description

rndc.conf is the configuration file for *rndc(1)*, the remote control program for *named(8)*.

5.2.2 Configuration grammar

The configuration file consists of configuration statements and comments. Statements end with a semicolon. Statements and comments are the only elements that can appear without enclosing braces. Many statements contain a block of sub-statements, which are also terminated with a semicolon. Clauses in the statements are also semi-colon terminated. See the [Comments syntax](#) section for a description of comments, and the [Examples](#) section for examples.

rndc.conf has a similar structure and syntax to *named.conf(5)*, although *rndc.conf*'s grammar is much simpler than that of *named.conf(5)*. *rndc.conf* supports the following statements:

key

Defines a symbolic name for a control channel key of the specified algorithm and secret.

server

Defines a symbolic name for a nameserver, and includes configuration on how to communicate with it.

options

Defines default configuration values for ***rndc***.

5.2.2.1 **key** statement

```
key <string> {  
    algorithm <string>;  
    secret <string>;  
}; // may occur multiple times
```

The *key* statement begins with an identifying string, the name of the key. The statement has two clauses. *algorithm* identifies the authentication algorithm for ***rndc*** to

use `--- hmac-sha256` (default) and `hmac-sha512` are supported. This is followed by a secret clause which contains the Base64 encoding ([RFC 4648](#)) of the algorithm's authentication key. The Base64 string is enclosed in double quotes.

See the [Examples](#) section for a command to generate a sample `rndc.conf` file with a key statement.

5.2.2.2 server statement

```
server <string> {
    key <string>;
    port <integer>;
    source-address ( <ipv4_address> | * );
    source-address-v6 ( <ipv6_address> | * );
    addresses { ( <quoted_string> [ port <integer> ] [
↳ dscp <integer> ] | <ipv4_address> [ port <integer> ] [ dscp
↳ <integer> ] | <ipv6_address> [ port <integer> ] [ dscp
↳ <integer> ] ); ... };
}; // may occur multiple times
```

After the *server* keyword, the server statement contains a string which is the hostname or address for a nameserver. The statement has three possible clauses: *key*, *port* and *addresses*. The key name must match the name of a key statement in the file. The port number specifies the port to connect to. If an *addresses* clause is supplied these addresses will be used instead of the server name. Each address can take an optional port. If an *source-address* or *source-address-v6* of supplied then these will be used to specify the IPv4 and IPv6 source addresses respectively.

5.2.2.3 options statement

```
options {
    default-key <string>;
    default-port <integer>;
    default-server <string>;
    default-source-address ( <ipv4_address> | * );
    default-source-address-v6 ( <ipv6_address> | * );
};
```

The *options* statement contains the following clauses:

- The *default-server* clause is followed by the name or address of the **named** nameserver. This host will be used when no nameserver is given as an argument to **rndc**.
- The *default-key* clause is followed by the name of a key which is identified by a *key* statement. If no *keyid* is provided on the **rndc** command line, and no *key* clause is found in a matching *server* statement, this default key will be used to authenticate the nameserver's commands and responses.
- The *default-port* clause is followed by the port number to connect to on the remote

nameserver. If no *port* option is provided on the **rndc** command line, and no *port* clause is found in a matching *server* statement, this default port number will be connected to.

- The *default-source-address* clause is used to set the source IPv4 address.
- The *default-source-address-v6* clause is used to set the source IPv6 address.

Note

The `options` statement may only occur once in the configuration file.

5.2.3 Comments syntax

The comment syntax allows for comments to appear anywhere that whitespace may appear in a config file. To appeal to all programmers, they can be written in the C, C++, or shell/Perl style:

- C-style comments start with the two characters `/*` (slash, star) and end with `*/` (star, slash). Because they are completely delimited within these characters, they can be used to comment only a portion of a line or to span multiple lines. For example:

```
/*
 * This is a multi-line
 * comment.
 */
```

C-style comments cannot be nested. For example, the following syntax is not valid because the entire comment ends with the first `*/`:

```
/* This is the start of a comment.
   This is still part of the comment.
/* This is an incorrect attempt at nesting a comment. */
   This is no longer in any comment. */
```

- C++-style comments start with the two characters `//` (slash, slash) and continue to the end of the physical line. They cannot be continued across multiple physical lines; to have one logical comment span multiple lines, each line must use the `//` pair. For example:

```
// This is a comment. It continues to end of line.
// This next line is a new comment, even though it is
// logically part of the previous comment.
```

- Shell-style or Perl-style comments start with the character `#` (number/hash sign) and continue to the end of the physical line. They cannot be continued across multiple physical lines; to have one logical comment span multiple lines, each line must use the `#` character. For example:

```
# This is a comment. It continues to end of line.  
# This next line is a new comment, even though it is  
# logically part of the previous comment.
```

Note

You cannot use the ; (semi-colon) character to start a comment such as you would in a zone file. The semicolon indicates the end of a configuration statement.

5.2.4 Examples

A sample `rndc.conf` file follows:

```
options {  
    default-server    localhost;  
    default-key       samplekey;  
};  
  
server localhost {  
    key               samplekey;  
};  
  
server testserver {  
    key              testkey;  
    addresses       { localhost port 5353; };  
};  
  
key samplekey {  
    algorithm         hmac-sha256;  
    secret            "6FMfj430sz4lyb240Ie2iGEz9lf1llJO+lz";  
};  
  
key testkey {  
    algorithm         hmac-sha256;  
    secret            "R3HI8P6BKw9ZwXwN3VZKuQ==";  
};
```

In the above sample, **`rndc`** will by default use the server *localhost* and the key called *samplekey*. Commands to the *localhost* server will use the *samplekey* key, which must also be defined in the server's configuration file with the same name and secret. The **key** statement indicates that *samplekey* uses the `hmac-sha256` algorithm and its **secret** clause contains the Base64 encoding of the `hmac-sha256` secret enclosed in double quotes.

If **`rndc`** is then run with `-s testserver` arguments, then **`rndc`** will connect to the nameserver on *localhost* port 5353 using the key *testkey*.

To generate a sample `rndc.conf`, run **`rndc-confgen`** without arguments:

```

$ rndc-confgen
# Start of rndc.conf
key "rndc-key" {
    algorithm hmac-sha256;
    secret "6ttb0LmEkplPXfQ6wvcnwWUYhBfGlPjTwFA9hOYrluw=";
};

options {
    default-key "rndc-key";
    default-server 127.0.0.1;
    default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow_
↪list as needed:
# key "rndc-key" {
#     algorithm hmac-sha256;
#     secret "6ttb0LmEkplPXfQ6wvcnwWUYhBfGlPjTwFA9hOYrluw=";
# };
#
# controls {
#     inet 127.0.0.1 port 953
#         allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
$

```

The content for a complete `rndc.conf` file, including a key statement with a randomly generated secret, will be written to the standard output. Commented-out **key** and **controls** statements for `named.conf` (5) are also printed.

5.2.5 Nameserver configuration

`named` (8) must be configured to accept control-channel connections and to recognize the key specified in the `rndc.conf` file, using the **controls** statement in `named.conf` (5).

5.2.6 See also

`named` (8), `named.conf` (5), `rndc` (1), `rndc-confgen` (1)

5.2.7 Copyright

Copyright (C) 2024 Banu Systems Private Limited. All rights reserved.

Copyright (c) 2001, 2004-2005, 2007, 2013-2016, 2018 Internet Systems Consortium, Inc. ("ISC").

FEATURES

6.1 Resolver

6.1.1 Smoothed round-trip time computation

The Loop resolver maintains a smoothed round-trip time (SRTT) for every nameserver address in the ADB. The SRTT is used to select the next nameserver to contact when there are multiple nameservers available.

It is initialized and updated during a fetch operation as follows:

1. Initially the SRTT to a nameserver address is set to a random value between [0, 31] microseconds, so that the nameserver can be tried in some random order as no RTT is currently known for that address.
2. Whenever a response is received from a nameserver address, an RTT from that roundtrip is available to Loop. The SRTT for that nameserver address is updated as 7/10ths of the old SRTT plus 3/10ths of the RTT of the response. Hence the term "smoothed".
3. Whenever a request to a nameserver address ends in timeout and there's no response, the RTT for that request is not available. In this case:
 - (a) In the case of EDNS queries for which no EDNS response has been received from that address previously, the SRTT is not updated, as the response may not have been received due to improper support for EDNS and not actual network conditions.
 - (b) In all other cases, the SRTT for that nameserver address is increased as follows, and then clamped to a maximum of the maximum single query timeout value which is 9 seconds:
 - (i) If the existing SRTT value is greater than 800000 microseconds, increase by a random value in the interval [0, 16383] microseconds.
 - (ii) If the existing SRTT value is in the half-open interval (400000, 800000] microseconds, increase by a random value in the interval [0, 32767] microseconds.
 - (iii) If the existing SRTT value is in the half-open interval (200000, 400000]

microseconds, increase by a random value in the interval [0, 65535] microseconds.

- (iv) If the existing SRTT value is in the half-open interval (100000, 200000] microseconds, increase by a random value in the interval [0, 131071] microseconds.
 - (v) If the existing SRTT value is in the half-open interval (50000, 100000] microseconds, increase by a random value in the interval [0, 262143] microseconds.
 - (vi) If the existing SRTT value is in the half-open interval (25000, 50000] microseconds, increase by a random value in the interval [0, 524287] microseconds.
 - (vii) If the existing SRTT value is less than or equal to 25000 microseconds, increase by a random value in the interval [0, 1048575] microseconds.
4. For all other nameservers in that fetch that were not contacted, and that are not marked as bad or marked for some other reason, their SRTTs are aged by multiplying them with a factor 511/512 using integer arithmetic (~0.998046875) which lowers their SRTT slightly.

6.1.2 Nameserver selection

For nameserver selection, Loop initially tries to gather a variety of nameserver addresses to be used with the fetch depending on the view configuration and the nameservers for the current zonecut. Some of these are skipped if they are known to be bad. The lists may not be necessarily complete too depending on what addresses are known currently. Finally, the lists of nameserver addresses that we have are sorted by their SRTT values to pick the order in which they're contacted.

Error

TODO: This has to be described in more detail.

6.2 DNS NOTIFY

DNS NOTIFY (see [RFC 1996](#)) is a mechanism that allows master servers to notify their slave servers of changes to a zone's data. When a slave receives a NOTIFY message from a master server, it will check to see that its version of the zone is the current version and, if not, initiate a zone transfer.

For more information about DNS NOTIFY, see the description of the `notify` option in *section_title* and the description of the zone option `also-notify` in *section_title*. The NOTIFY protocol is specified in [RFC 1996](#).

Note

As a slave zone can also be a master to other slaves, `named`, by default, sends DNS NOTIFY messages for every zone it loads. Specifying `notify master-only` will cause `named` to only send NOTIFY for master zones that it loads.

6.3 DNS UPDATE (dynamic updates)

DNS UPDATE (see [RFC 2136](#)) is a method for adding, replacing or deleting records in a master server by sending it a special form of DNS messages. This method is also referred to as *dynamic update*.

Dynamic update is enabled by including an `allow-update` or an `update-policy` clause in the zone statement.

If the zone's `update-policy` is set to `local`, updates to the zone will be permitted for the key `local-ddns`, which will be generated by `named` at startup. See *section_title* for more details.

Dynamic updates using Kerberos signed requests can be made using the TKEY/GSS protocol by setting either the `tkey-gssapi-keytab` option, or alternatively by setting both the `tkey-gssapi-credential` and `tkey-domain` options. Once enabled, Kerberos signed requests will be matched against the update policies for the zone, using the Kerberos principal as the signer for the request.

Updating of secure zones (zones using DNSSEC) follows RFC 3007: RRSIG, NSEC and NSEC3 records affected by updates are automatically regenerated by the server using an online zone key. Update authorization is based on transaction signatures and an explicit server policy.

6.3.1 Journal files

All changes made to a zone using dynamic update are stored in the zone's journal file. This file is automatically created by the server when the first dynamic update takes place. The name of the journal file is formed by appending the extension `.jnl` to the name of the corresponding zone file unless specifically overridden. The journal file is in a binary format and should not be edited manually.

The server will also occasionally write ("dump") the complete contents of the updated zone to its zone file. This is not done immediately after each dynamic update, because that would be too slow when a large zone is updated frequently. Instead, the dump is delayed by up to 15 minutes, allowing additional updates to take place. During the dump process, transient files will be created with the extensions `.jnw` and `.jbnk`; under ordinary circumstances, these will be removed when the dump is complete, and can be safely ignored.

When a server is restarted after a shutdown or crash, it will replay the journal file to incorporate into the zone any updates that took place after the last zone dump.

Changes that result from incoming incremental zone transfers are also journalled in a similar way.

The zone files of dynamic zones cannot normally be edited by hand because they are not guaranteed to contain the most recent dynamic changes — those are only in the journal file. The only way to ensure that the zone file of a dynamic zone is up to date is to run `rndc stop`.

If you have to make changes to a dynamic zone manually, the following procedure will work: Disable dynamic updates to the zone using `rndc freeze zone`. This will update the zone's master file with the changes stored in its `.jnl` file. Edit the zone file. Run `rndc thaw zone` to reload the changed zone and re-enable dynamic updates.

`rndc sync zone` will update the zone file with changes from the journal file without stopping dynamic updates; this may be useful for viewing the current zone state. To remove the `.jnl` file after updating the zone file, use `rndc sync -clean`.

6.4 IXFR (incremental zone transfers)

The incremental zone transfer (IXFR) protocol (see [RFC 1995](#)) is a way for slave servers to transfer only changed data, instead of having to transfer the entire zone.

When acting as a master, Loop supports IXFR for those zones where the necessary change history information is available. These include master zones maintained by dynamic update and slave zones whose data was obtained by IXFR. For manually maintained master zones, and for slave zones obtained by performing a full zone transfer (AXFR), IXFR is supported only if the option `ixfr-from-differences` is set to `yes`.

When acting as a slave, Loop will attempt to use IXFR unless it is explicitly disabled. For more information about disabling IXFR, see the description of the `request-ixfr` clause of the `server` statement.

6.5 Split DNS

Setting up different views, or visibility, of the DNS space to internal and external resolvers is usually referred to as a *Split DNS* setup. There are several reasons an organization would want to set up its DNS this way.

One common reason for setting up a DNS system this way is to hide "internal" DNS information from "external" clients on the Internet. There is some debate as to whether or not this is actually useful. Internal DNS information leaks out in many ways (via email headers, for example) and most savvy "attackers" can find the information they need using other means. However, since listing addresses of internal servers that external clients cannot possibly reach can result in connection delays and other annoyances, an organization may choose to use a Split DNS to present a consistent view of itself to the outside world.

Another common reason for setting up a Split DNS system is to allow internal networks that are behind filters or in RFC 1918 space (reserved IP space, as documented in RFC 1918) to resolve DNS on the Internet. Split DNS can also be used to allow mail from outside back in to the internal network.

6.5.1 Example split DNS setup

Let's say a company named *Example, Inc.* (`example.com`) has several corporate sites that have an internal network with reserved Internet Protocol (IP) space and an external demilitarized zone (DMZ), or "outside" section of a network, that is available to the public.

Example, Inc. wants its internal clients to be able to resolve external hostnames and to exchange mail with people on the outside. The company also wants its internal resolvers to have access to certain internal-only zones that are not available at all outside of the internal network.

In order to accomplish this, the company will set up two sets of name servers. One set will be on the inside network (in the reserved IP space) and the other set will be on bastion hosts, which are "proxy" hosts that can talk to both sides of its network, in the DMZ.

The internal servers will be configured to forward all queries, except queries for `site1.internal`, `site2.internal`, `site1.example.com`, and `site2.example.com`, to the servers in the DMZ. These internal servers will have complete sets of information for `site1.example.com`, `site2.example.com`, `site1.internal`, and `site2.internal`.

To protect the `site1.internal` and `site2.internal` domains, the internal name servers must be configured to disallow all queries to these domains from any external hosts, including the bastion hosts.

The external servers, which are on the bastion hosts, will be configured to serve the "public" version of the `site1` and `site2.example.com` zones. This could include things such as the host records for public servers (`www.example.com` and `ftp.example.com`), and mail exchange (MX) records (`a.mx.example.com` and `b.mx.example.com`).

In addition, the public `site1` and `site2.example.com` zones should have special MX records that contain wildcard (*) records pointing to the bastion hosts. This is needed because external mail servers do not have any other way of looking up how to deliver mail to those internal hosts. With the wildcard records, the mail will be delivered to the bastion host, which can then forward it on to internal hosts.

Here's an example of a wildcard MX record:

```
*      IN MX 10 external1.example.com.
```

Now that they accept mail on behalf of anything in the internal network, the bastion hosts will need to know how to deliver mail to internal hosts. In order for this to work properly, the resolvers on the bastion hosts will need to be configured to point to the internal name servers for DNS resolution.

Queries for internal hostnames will be answered by the internal servers, and queries for external hostnames will be forwarded back out to the DNS servers on the bastion hosts.

In order for all this to work properly, internal clients will need to be configured to

query *only* the internal name servers for DNS queries. This could also be enforced via selective filtering on the network.

If everything has been set properly, *Example, Inc.*'s internal clients will now be able to:

- Look up any hostnames in the `site1` and `site2.example.com` zones.
- Look up any hostnames in the `site1.internal` and `site2.internal` domains.
- Look up any hostnames on the Internet.
- Exchange mail with both internal and external people.

Hosts on the Internet will be able to:

- Look up any hostnames in the `site1` and `site2.example.com` zones.
- Exchange mail with anyone in the `site1` and `site2.example.com` zones.

Here is an example configuration for the setup we just described above. Note that this is only configuration information; for information on how to configure your zone files, see *section_title*.

Internal DNS server config:

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };

options {
    ...
    ...
    forward only;
    // forward to external servers
    forwarders {
        bastion-ips-go-here;
    };
    // sample allow-transfer (no one)
    allow-transfer { none; };
    // restrict query access
    allow-query { internals; externals; };
    // restrict recursion
    allow-recursion { internals; };
    ...
    ...
};

// sample master zone
zone "site1.example.com" {
    type master;
    file "m/site1.example.com";
```

(continues on next page)

(continued from previous page)

```
// do normal iterative resolution (do not forward)
forwarders { };
allow-query { internals; externals; };
allow-transfer { internals; };
};

// sample slave zone
zone "site2.example.com" {
    type slave;
    file "s/site2.example.com";
    masters { 172.16.72.3; };
    forwarders { };
    allow-query { internals; externals; };
    allow-transfer { internals; };
};

zone "site1.internal" {
    type master;
    file "m/site1.internal";
    forwarders { };
    allow-query { internals; };
    allow-transfer { internals; };
};

zone "site2.internal" {
    type slave;
    file "s/site2.internal";
    masters { 172.16.72.3; };
    forwarders { };
    allow-query { internals; };
    allow-transfer { internals; };
};
```

External (bastion host) DNS server config:

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };

options {
    ...
    ...
    // sample allow-transfer (no one)
    allow-transfer { none; };
    // default query access
    allow-query { any; };
}
```

(continues on next page)

(continued from previous page)

```
// restrict cache access
allow-query-cache { internals; externals; };
// restrict recursion
allow-recursion { internals; externals; };
...
...
};

// sample slave zone
zone "site1.example.com" {
    type master;
    file "m/site1.foo.com";
    allow-transfer { internals; externals; };
};

zone "site2.example.com" {
    type slave;
    file "s/site2.foo.com";
    masters { another_bastion_host_maybe; };
    allow-transfer { internals; externals; };
};
```

In the `resolv.conf` (or equivalent) on the bastion host(s):

```
search ...
nameserver 172.16.72.2
nameserver 172.16.72.3
nameserver 172.16.72.4
```

6.5.2 Example split DNS setup 2

Here is an example of a typical split DNS setup implemented using `view` statements:

```
view "internal" {
    // This should match our internal networks.
    match-clients { 10.0.0.0/8; };

    // Provide recursive service to internal
    // clients only.
    recursion yes;

    // Provide a complete view of the example.com
    // zone including addresses of internal hosts.
    zone "example.com" {
        type master;
        file "example-internal.db";
    };
};
```

(continues on next page)

(continued from previous page)

```

    };
};

view "external" {
    // Match all clients not matched by the
    // previous view.
    match-clients { any; };

    // Refuse recursive service to external clients.
    recursion no;

    // Provide a restricted view of the example.com
    // zone containing only publicly accessible hosts.
    zone "example.com" {
        type master;
        file "example-external.db";
    };
};

```

6.6 TSIG (transaction signatures)

TSIG (originally specified in [RFC 2845](#), extended in [RFC 4635](#), and revised in [RFC 8945](#)) is a mechanism for authenticating DNS messages between two parties by cryptographically signing them using HMACs ([RFC 2104](#)). TSIG can be used in any DNS transaction as a way to restrict access to certain nameserver functions (e.g., recursive queries) only to authorized clients, when IP-based access control is insufficient or needs to be overridden, or as a way to ensure message authenticity when it is critical to the integrity of data, such as with DNS UPDATE messages or zone transfers from a primary to a secondary nameserver.

This is a guide to setting up TSIG in Loop. It describes the configuration syntax and the process of creating TSIG keys.

named supports TSIG for server-to-server communication, and some of the tools included with Loop support TSIG for sending messages to **named**:

- **dig** supports TSIG via the `-k` and `-y` command line options.
- **nsupdate** supports TSIG via the `-k`, `-l`, and `-y` command line options, or via the `key` command when running interactively.

6.6.1 Generating a shared key

TSIG keys can be generated using the `ddns-confgen(1)` program; the output of the command is a key directive suitable for inclusion in `named.conf`. The key name, algorithm and size can be specified by command line parameters; the defaults are "ddns-key", HMAC-SHA256, and 256 bits, respectively.

Any string which is a valid DNS name can be used as a key name. For example, a key to be shared between servers called *host1* and *host2* could be called "host1-host2.", and this key could be generated using:

```
$ ddns-confgen -q -k host1-host2. > host1-host2.key
```

This key may then be copied to both hosts. The key name and secret must be identical on both hosts.

Warning

Copying a shared secret from one server to another is beyond the scope of the DNS. A secure transport mechanism should be used.

When run without the **-q** option, **ddns-confgen**'s output includes additional configuration text for setting up dynamic DNS in *named(8)*. See *ddns-confgen(1)* for details.

6.6.2 Loading a new key

For a key shared between servers called *host1* and *host2*, the following could be added to each server's *named.conf* file:

```
key "host1-host2." {  
    algorithm hmac-sha256;  
    secret "DAopyf1mhCbFVZw7pgmNPBoLUq8wEUT7UuPoLENP2HY=";  
};
```

(This is the same key generated above using **ddns-confgen**.)

Since this text contains a secret, it is recommended that either *named.conf* not be world-readable, or that the *key* directive be stored in a file which is not world-readable, and which is included in *named.conf* via the *include* directive.

Once a key has been added to *named.conf* and the server has been restarted or re-configured, the server can recognize the key. If the server receives a message signed by the key, it will be able to verify the signature. If the signature is valid, the response will be signed using the same key.

TSIG keys that are known to a server can be listed using the command *rndc tsig-list*.

6.6.3 Instructing the server to use a key

A server sending a request to another server must be told whether to use a key, and if so, which key to use.

For example, a key may be specified for each server in the *masters* statement in the definition of a slave zone; in this case, all SOA QUERY messages, NOTIFY messages, and zone transfer requests (AXFR or IXFR) will be signed using the specified key.

Keys may also be specified in the `also-notify` statement of a master or slave zone, causing NOTIFY messages to be signed using the specified key.

Keys can also be specified in a `server` directive. Adding the following on *host1*, if the IP address of *host2* is 10.1.2.3, would cause *all* requests from *host1* to *host2*, including normal DNS queries, to be signed using the `host1-host2.` key:

```
server 10.1.2.3 {  
    keys { host1-host2. ; }  
};
```

Multiple keys may be present in the `keys` statement, but only the first one is used. As this directive does not contain secrets, it can be used in a world-readable file.

Requests sent by *host2* to *host1* would *not* be signed, unless a similar `server` directive were in *host2*'s configuration file.

Whenever any server sends a TSIG-signed DNS request, it will expect the response to be signed with the same key. If a response is not signed, or if the signature is not valid, the response will be rejected.

6.6.4 TSIG-based access control

TSIG keys may be specified in ACL definitions and ACL directives such as `allow-query`, `allow-transfer` and `allow-update`. The above key would be denoted in an ACL element as `key host1-host2.`

An example of an `allow-update` directive using a TSIG key:

```
allow-update { !{ !localnets; any; }; key host1-host2. ; };
```

This allows dynamic updates to succeed only if the UPDATE request comes from an address in `localnets`, *and* if it is signed using the `host1-host2.` key.

See *section_title* for a discussion of the more flexible `update-policy` statement.

6.6.5 TSIG errors

Processing of TSIG-signed messages can result in several kinds of errors:

- If a TSIG-aware server receives a message signed by an unknown key, the response will be unsigned, with the TSIG extended error code set to BADKEY.
- If a TSIG-aware server receives a message from a known key but with an invalid signature, the response will be unsigned, with the TSIG extended error code set to BADSIG.
- If a TSIG-aware server receives a message with a time outside of the allowed range, the response will be signed, with the TSIG extended error code set to BADTIME, and the time values will be adjusted so that the response can be successfully verified.

In all of the above cases, the server will return a response code of NOTAUTH (not authenticated).

6.7 TKEY (transaction keys)

TKEY (originally specified in [RFC 2930](#)) is a mechanism for automatically negotiating a TSIG shared secret between two hosts.

There are several TKEY "modes" that specify how a key is to be generated or assigned. Loop implements only one of these modes: Diffie-Hellman key exchange. Both hosts are required to have a KEY record with algorithm DH (though this record is not required to be present in a zone).

The TKEY process is initiated by a client or server by sending a query of type TKEY to a TKEY-aware server. The query must include an appropriate KEY record in the additional section, and must be signed using either TSIG or SIG(0) with a previously established key. The server's response, if successful, will contain a TKEY record in its answer section. After this transaction, both participants will have enough information to calculate a shared secret using Diffie-Hellman key exchange. The shared secret can then be used by to sign subsequent transactions between the two servers.

TSIG keys known by the server, including TKEY-negotiated keys, can be listed using `rndc tsig-list`.

TKEY-negotiated keys can be deleted from a server using `rndc tsig-delete`. This can also be done via the TKEY protocol itself, by sending an authenticated TKEY query specifying the "key deletion" mode.

Whenever **named** is shutdown, for each view it dumps keyrings containing dynamically negotiated keys to a file respectively; the filename will be the view name followed by the suffix `.tsigkeys`. Whenever **named** is restarted, it attempts to restore any dumped keyrings from previous invocations of **named**.

Error

TODO: This should be replaced with the hashed filename mechanism.

6.8 SIG(0)

Loop partially supports SIG(0) transaction signatures as specified in [RFC 2535](#) and [RFC 2931](#). SIG(0) uses public/private keys to authenticate messages. Access control is performed in the same manner as TSIG keys; privileges can be granted or denied in ACL directives based on the key name.

When a SIG(0) signed message is received, it will only be verified if the key is known and trusted by the server. The server will not attempt to recursively fetch or validate the key.

SIG(0) signing of multiple-message TCP streams is not supported.

nsupdate is the only program shipped with Loop that generates SIG(0) signed messages.

6.9 DNSSEC

Cryptographic authentication of DNS information is possible through the DNS Security (*DNSSEC-bis*) extensions, defined in RFC 4033, RFC 4034, and RFC 4035. This section describes the creation and use of DNSSEC signed zones.

In order to set up a DNSSEC secure zone, there are a series of steps which must be followed. Loop ships with several tools that are used in this process, which are explained in more detail below. In all cases, the `-h` option prints a full list of parameters. Note that the DNSSEC tools require the keyset files to be in the working directory or the directory specified by the `-d` option.

There must also be communication with the administrators of the parent and/or child zone to transmit keys. A zone's security status must be indicated by the parent zone for a DNSSEC capable resolver to trust its data. This is done through the presence or absence of a DS record at the delegation point.

For other servers to trust data in this zone, they must either be statically configured with this zone's zone key or the zone key of another zone above this one in the DNS tree.

6.9.1 Generating Keys

The *dnssec-keygen* (1) program is used to generate keys.

A secure zone must contain one or more zone keys. The zone keys will sign all other records in the zone, as well as the zone keys of any secure delegated zones. Zone keys must have the same name as the zone, a name type of `ZONE`, and must be usable for authentication. It is recommended that zone keys use a cryptographic algorithm designated as "mandatory to implement" by the IETF; currently the only one is RSASHA1.

The following command will generate a 2048-bit RSASHA256 key for the `child.example` zone:

```
dnssec-keygen -a RSASHA256 -b 2048 -n ZONE child.example.
```

Two output files will be produced: `Kchild.example.+008+12345.key` and `Kchild.example.+008+12345.private` (where 12345 is an example of a key tag). The key filenames contain the key name (`child.example.`), algorithm (5 is RSASHA1, 8 is RSASHA256, 13 is ECDSAP256SHA256, etc.), and the key tag (12345 in this case). The private key (in the `.private` file) is used to generate signatures, and the public key (in the `.key` file) is used for signature verification.

To generate another key with the same properties (but with a different key tag), repeat the above command.

The *dnssec-keyfromlabel* (1) program can be used to get a key pair from a crypto hardware and build the key files. It can be used similar to *dnssec-keygen* (1).

The public keys should be inserted into the zone file by including the `.key` files using `$INCLUDE` statements.

6.9.2 Signing the Zone

The `dnssec-signzone` program is used to sign a zone.

Any `keyset` files corresponding to secure subzones should be present. The zone signer will generate `NSEC`, `NSEC3` and `RRSIG` records for the zone, as well as `DS` for the child zones if `'-g'` is specified. If `'-g'` is not specified, then `DS` `RRsets` for the secure child zones need to be added manually.

The following command signs the zone, assuming it is in a file called `zone.child.example`. By default, all zone keys which have an available private key are used to generate signatures.

```
dnssec-signzone -o child.example zone.child.example
```

One output file is produced: `zone.child.example.signed`. This file should be referenced by `named.conf` as the input file for the zone.

`dnssec-signzone` will also produce a `keyset` and `dsset` files and optionally a `dlvset` file. These are used to provide the parent zone administrators with the `DNSKEYs` (or their corresponding `DS` records) that are the secure entry point to the zone.

6.9.3 Configuring Servers

To enable `named` to respond appropriately to DNS requests from DNSSEC aware clients, `dnssec-enable` must be set to `yes`. (This is the default setting.)

To enable `named` to validate answers from other servers, the `dnssec-enable` option must be set to `yes`, and the `dnssec-validation` options must be set to `yes` or `auto`.

If `dnssec-validation` is set to `auto`, then a default trust anchor for the DNS root zone will be used. If it is set to `yes`, however, then at least one trust anchor must be configured with a `trusted-keys` or `managed-keys` statement in `named.conf`, or DNSSEC validation will not occur. The default setting is `yes`.

`trusted-keys` are copies of `DNSKEY` `RRs` for zones that are used to form the first link in the cryptographic chain of trust. All keys listed in `trusted-keys` (and corresponding zones) are deemed to exist and only the listed keys will be used to validate the `DNSKEY` `RRset` that they are from.

`managed-keys` are trusted keys which are automatically kept up to date via RFC 5011 trust anchor maintenance.

`trusted-keys` and `managed-keys` are described in more detail later in this document.

Loop does not verify signatures on load, so zone keys for authoritative zones do not need to be specified in the configuration file.

After DNSSEC gets established, a typical DNSSEC configuration will look something like the following. It has one or more public keys for the root. This allows answers

from outside the organization to be validated. It will also have several keys for parts of the namespace the organization controls. These are here to ensure that named is immune to compromises in the DNSSEC components of the security of parent zones.

```
managed-keys {
    /* Root Key */
    "." initial-key 257 3 3
    ↪ "BNY4wrWMlnCfJ+CXd0rVXyYmobt7sEEfK3clRbGaTwS
        JxrGkxJWoZu6I7PzJu/E9gx4UC1zGAHlXKdE4zYIpRh
        aBKnvcC2U9mZhkdUpd1Vso/HAdjNe8LmMlnzY3zy2Xy
        4klWOADTPzSv9eamj8Vl8PHGjBLaVtYvk/ln5ZApjYg
        hf+6fElrmLkdaz MQ2OCnACR817DF4BBa7UR/beDHyp
        5iWTXWSi6XmoJLbG9Scqc7l70KDqlvXR3M/1UUVRbke
        glIPJSidmK3ZyCllh4XSKbje/45SKucHgnwU5jefMtq
        66gKodQj+MiA21AfUve7u99WzTLzY3qlxDhxYQQ20FQ
        97S+LKUTpQcq27R7AT3/V5hRQxScINqwcZ4jYqZD2fQ
        dgxbcDTC1U0CRBdiieyLMNzXG3";
};

trusted-keys {
    /* Key for our organization's forward zone */
    example.com. 257 3 5 "AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM6
        5KbhTjrWlZaARmPhEZZe3Y9ifgEuq7vZ/z
        GZUdEGNWy+JZzus0lUptwgjGwhUS1558Hb
        4JKUbbOTcM8pwXlj0EiX3oDFVmjhO444gL
        kBOUKUf/mC7HvfwYH/Be22GnClrinKJp1O
        g4yWzO9WglMk7jbfW33gUKvirTHr25GL7S
        TQUzBb5Usxt8lgnyTUHs1t3JwCY5hKZ6Cq
        FxmAVZP20igTixin/1LcrgX/KMEGd/biuv
        F4qJCyduieHukuY3H4XMAcR+xia2nIUPvm
        /oyWR8BW/hWdzOvnSCThlHf3xiYleDbt/o
        1OTQ09A0=";

    /* Key for our reverse zone. */
    2.0.192.IN-ADDRPA.NET. 257 3 5 "AQOnS4xn/IgOUpBPJ3bogzwc
        xOdNax071L18QqZnQQQAVVr+i
        LhGTnNGp3HoWQLUIzKrJVZ3zg
        gy3WwNT6kZo6c0tszYqbtvchm
        gQC8CzKojM/W16i6MG/eafGU3
        siaOdS0yOI6BgPsw+YZdzlYMa
        IJGf4M4dyoKIhZdZyQ2bYQrjy
        Q4LB0lC7aOnsMyYKHHYeRvPxj
        IQXmdqgOJGq+vsevG06zW+1xg
        YJh9rCIfnm1GX/KMgxLPG2vXT
        D/RnLX+D3T3UL7HJYHJhAZD5L
        59VvjSPsZJHeDCUyWYrvPZesZ
        DIRvhDD52SKvbheeTJU6Ehkz
```

(continues on next page)

(continued from previous page)

```
ytNN2SN96QRk8j/iI8ib";  
};  
  
options {  
    ...  
    dnssec-enable yes;  
    dnssec-validation yes;  
};
```

Note

None of the keys listed in this example are valid. In particular, the root key is not valid.

When DNSSEC validation is enabled and properly configured, the resolver will reject any answers from signed, secure zones which fail to validate, and will return SERV-FAIL to the client.

Responses may fail to validate for any of several reasons, including missing, expired, or invalid signatures, a key which does not match the DS RRset in the parent zone, or an insecure response from a zone which, according to its parent, should have been secure.

Note

When the validator receives a response from an unsigned zone that has a signed parent, it must confirm with the parent that the zone was intentionally left unsigned. It does this by verifying, via signed and validated NSEC/NSEC3 records, that the parent zone contains no DS records for the child.

If the validator *can* prove that the zone is insecure, then the response is accepted. However, if it cannot, then it must assume an insecure response to be a forgery; it rejects the response and logs an error.

The logged error reads "insecurity proof failed" and "got insecure response; parent indicates it should be secure".

6.10 DNSSEC, Dynamic Zones, and Automatic Signing

6.10.1 Converting from insecure to secure

Changing a zone from insecure to secure can be done in two ways: using a dynamic DNS update, or the `auto-dnssec` zone option.

For either method, you need to configure `named` so that it can see the `K*` files which contain the public and private parts of the keys that will be used to sign the zone. These files will have been generated by `dnssec-keygen`. You can do this by placing them in the `key-directory`, as specified in `named.conf`:

```
zone example.net {
    type master;
    update-policy local;
    file "dynamic/example.net/example.net";
    key-directory "dynamic/example.net";
};
```

If one KSK and one ZSK DNSKEY key have been generated, this configuration will cause all records in the zone to be signed with the ZSK, and the DNSKEY RRset to be signed with the KSK as well. An NSEC chain will be generated as part of the initial signing process.

6.10.2 Dynamic DNS update method

To insert the keys via dynamic update:

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7
  ↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8a135zz3Y
  ↪I1m/SAQBxIqMfLtIwqWPDgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAAd/7odU/
  ↪64o2LGsifbLtQmtO8dFDtTAZXsX2+X3e/UNlq9IHq3Y0 XtC0Iuawl/
  ↪qkaKVxXe2lo8Ct+dM6UehyCqk=
> send
```

While the update request will complete almost immediately, the zone will not be completely signed until `named` has had time to walk the zone and generate the NSEC and RRSIG records. The NSEC record at the apex will be added last, to signal that there is a complete NSEC chain.

If you wish to sign using NSEC3 instead of NSEC, you should add an NSEC3PARAM record to the initial update request. If you wish the NSEC3 chain to have the OPTOUT bit set, set it in the flags field of the NSEC3PARAM record.

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7
  ↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8a135zz3Y
  ↪I1m/SAQBxIqMfLtIwqWPDgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAAd/7odU/
  ↪64o2LGsifbLtQmtO8dFDtTAZXsX2+X3e/UNlq9IHq3Y0 XtC0Iuawl/
  ↪qkaKVxXe2lo8Ct+dM6UehyCqk=
> update add example.net NSEC3PARAM 1 1 100 1234567890
> send
```

Again, this update request will complete almost immediately; however, the record won't show up until `named` has had a chance to build/remove the relevant chain. A

private type record will be created to record the state of the operation (see below for more details), and will be removed once the operation completes.

While the initial signing and NSEC/NSEC3 chain generation is happening, other updates are possible as well.

6.10.3 Fully automatic zone signing

To enable automatic signing, add the `auto-dnssec` option to the zone statement in `named.conf`. `auto-dnssec` has two possible arguments: `allow` or `maintain`.

With `auto-dnssec allow`, `named` can search the key directory for keys matching the zone, insert them into the zone, and use them to sign the zone. It will do so only when it receives an `rndc sign <zonename>`.

`auto-dnssec maintain` includes the above functionality, but will also automatically adjust the zone's DNSKEY records on schedule according to the keys' timing metadata. See the *`dnssec-keygen(1)`* and *`dnssec-settime(1)`* manpages for more information.

`named` will periodically search the key directory for keys matching the zone, and if the keys' metadata indicates that any change should be made the zone, such as adding, removing, or revoking a key, then that action will be carried out. By default, the key directory is checked for changes every 60 minutes; this period can be adjusted with the `dnssec-loadkeys-interval`, up to a maximum of 24 hours. The `rndc loadkeys` forces `named` to check for key updates immediately.

If keys are present in the key directory the first time the zone is loaded, the zone will be signed immediately, without waiting for an `rndc sign` or `rndc loadkeys` command. (Those commands can still be used when there are unscheduled key changes, however.)

When new keys are added to a zone, the TTL is set to match that of any existing DNSKEY RRset. If there is no existing DNSKEY RRset, then the TTL will be set to the TTL specified when the key was created (using the `dnssec-keygen -L` option), if any, or to the SOA TTL.

If you wish the zone to be signed using NSEC3 instead of NSEC, submit an NSEC3PARAM record via dynamic update prior to the scheduled publication and activation of the keys. If you wish the NSEC3 chain to have the OPTOUT bit set, set it in the flags field of the NSEC3PARAM record. The NSEC3PARAM record will not appear in the zone immediately, but it will be stored for later reference. When the zone is signed and the NSEC3 chain is completed, the NSEC3PARAM record will appear in the zone.

Using the `auto-dnssec` option requires the zone to be configured to allow dynamic updates, by adding an `allow-update` or `update-policy` statement to the zone configuration. If this has not been done, the configuration will fail.

6.10.4 Private-type records

The state of the signing process is signaled by private-type records (with a default type value of 65534). When signing is complete, these records will have a nonzero value for the final octet (for those records which have a nonzero initial octet).

The private type record format: If the first octet is non-zero then the record indicates that the zone needs to be signed with the key matching the record, or that all signatures that match the record should be removed.

algorithm (octet 1) key id in network order (octet 2 and 3) removal flag (octet 4) complete flag (octet 5)

Only records flagged as "complete" can be removed via dynamic update. Attempts to remove other private type records will be silently ignored.

If the first octet is zero (this is a reserved algorithm number that should never appear in a DNSKEY record) then the record indicates changes to the NSEC3 chains are in progress. The rest of the record contains an NSEC3PARAM record. The flag field tells what operation to perform based on the flag bits.

0x01 OPTOUT 0x80 CREATE 0x40 REMOVE 0x20 NONSEC

6.10.5 DNSKEY rollovers

As with insecure-to-secure conversions, rolling DNSSEC keys can be done in two ways: using a dynamic DNS update, or the `auto-dnssec` zone option.

6.10.6 Dynamic DNS update method

To perform key rollovers via dynamic update, you need to add the `K*` files for the new keys so that `named` can find them. You can then add the new DNSKEY RRs via dynamic update. `named` will then cause the zone to be signed with the new keys. When the signing is complete the private type records will be updated so that the last octet is non zero.

If this is for a KSK you need to inform the parent and any trust anchor repositories of the new KSK.

You should then wait for the maximum TTL in the zone before removing the old DNSKEY. If it is a KSK that is being updated, you also need to wait for the DS RRset in the parent to be updated and its TTL to expire. This ensures that all clients will be able to verify at least one signature when you remove the old DNSKEY.

The old DNSKEY can be removed via UPDATE. Take care to specify the correct key. `named` will clean out any signatures generated by the old key after the update completes.

6.10.7 Automatic key rollovers

When a new key reaches its activation date (as set by `dnssec-keygen` or `dnssec-settime`), if the `auto-dnssec zone` option is set to `maintain`, `named` will automatically carry out the key rollover. If the key's algorithm has not previously been used to sign the zone, then the zone will be fully signed as quickly as possible. However, if the new key is replacing an existing key of the same algorithm, then the zone will be re-signed incrementally, with signatures from the old key being replaced with signatures from the new key as their signature validity periods expire. By default, this rollover completes in 30 days, after which it will be safe to remove the old key from the DNSKEY RRset.

6.10.8 NSEC3PARAM rollovers via UPDATE

Add the new NSEC3PARAM record via dynamic update. When the new NSEC3 chain has been generated, the NSEC3PARAM flag field will be zero. At this point you can remove the old NSEC3PARAM record. The old chain will be removed after the update request completes.

6.10.9 Converting from NSEC to NSEC3

To do this, you just need to add an NSEC3PARAM record. When the conversion is complete, the NSEC chain will have been removed and the NSEC3PARAM record will have a zero flag field. The NSEC3 chain will be generated before the NSEC chain is destroyed.

6.10.10 Converting from NSEC3 to NSEC

To do this, use `nsupdate` to remove all NSEC3PARAM records with a zero flag field. The NSEC chain will be generated before the NSEC3 chain is removed.

6.10.11 Converting from secure to insecure

To convert a signed zone to unsigned using dynamic DNS, delete all the DNSKEY records from the zone apex using `nsupdate`. All signatures, NSEC or NSEC3 chains, and associated NSEC3PARAM records will be removed automatically. This will take place after the update request completes.

This requires the `dnssec-secure-to-insecure` option to be set to `yes` in `named.conf`.

In addition, if the `auto-dnssec maintain zone` statement is used, it should be removed or changed to `allow` instead (or it will re-sign).

6.10.12 Periodic re-signing

In any secure zone which supports dynamic updates, `named` will periodically re-sign RRsets which have not been re-signed as a result of some update action. The signature lifetimes will be adjusted so as to spread the re-sign load over time rather than all at once.

6.10.13 NSEC3 and OPTOUT

named only supports creating new NSEC3 chains where all the NSEC3 records in the zone have the same OPTOUT state. named supports UPDATES to zones where the NSEC3 records in the chain have mixed OPTOUT state. named does not support changing the OPTOUT state of an individual NSEC3 record, the entire chain needs to be changed if the OPTOUT state of an individual NSEC3 needs to be changed.

6.11 Dynamic Trust Anchor Management

Loop is able to maintain DNSSEC trust anchors using RFC 5011 key management. This feature allows named to keep track of changes to critical DNSSEC keys without any need for the operator to make changes to configuration files.

6.11.1 Validating Resolver

To configure a validating resolver to use RFC 5011 to maintain a trust anchor, configure the trust anchor using a `managed-keys` statement. Information about this can be found in the section titled *managed-keys statement*.

6.11.2 Authoritative Server

To set up an authoritative zone for RFC 5011 trust anchor maintenance, generate two (or more) key signing keys (KSKs) for the zone. Sign the zone with one of them; this is the "active" KSK. All KSK's which do not sign the zone are "stand-by" keys.

Any validating resolver which is configured to use the active KSK as an RFC 5011-managed trust anchor will take note of the stand-by KSKs in the zone's DNSKEY RRset, and store them for future reference. The resolver will recheck the zone periodically, and after 30 days, if the new key is still there, then the key will be accepted by the resolver as a valid trust anchor for the zone. Any time after this 30-day acceptance timer has completed, the active KSK can be revoked, and the zone can be "rolled over" to the newly accepted key.

The easiest way to place a stand-by key in a zone is to use the "smart signing" features of `dnssec-keygen` and `dnssec-signzone`. If a key with a publication date in the past, but an activation date which is unset or in the future, "`dnssec-signzone -S`" will include the DNSKEY record in the zone, but will not sign with it:

```
$ dnssec-keygen -K keys -f KSK -P now -A now+2y example.net
$ dnssec-signzone -S -K keys example.net
```

To revoke a key, the new command `dnssec-revoke` has been added. This adds the REVOKE bit (see [RFC 5011](#)) to the key flags and regenerates the `K*.key` and `K*.private` files.

After revoking the active key, the zone must be signed with both the revoked KSK and the new active KSK. (Smart signing takes care of this automatically.)

Once a key has been revoked and used to sign the DNSKEY RRset in which it appears, that key will never again be accepted as a valid trust anchor by the resolver. However, validation can proceed using the new active key (which had been accepted by the resolver when it was a stand-by key).

See RFC 5011 for more details on key rollover scenarios.

When a key has been revoked, its key ID changes, increasing by 128, and wrapping around at 65535. So, for example, the key "Kexample.com.+005+10000" becomes "Kexample.com.+005+10128".

If two keys have ID's exactly 128 apart, and one is revoked, then the two key ID's will collide, causing several problems. To prevent this, `dnssec-keygen` will not generate a new key if another key is present which may collide. This checking will only occur if the new keys are written to the same directory which holds all other keys in use for that zone.

It is expected that a future release of Loop will address this problem in a different way, by storing revoked keys with their original unrevoked key ID's.

6.12 PKCS#11

PKCS#11 (Public Key Cryptography Standard #11) defines a platform-independent API for the control of hardware security modules (HSMs) and other cryptographic support devices.

PKCS#11 makes use of a "provider library": a dynamically loadable library which provides a low-level PKCS#11 interface to drive the HSM hardware. The PKCS#11 provider library comes from the HSM vendor, and it is specific to the HSM to be controlled.

Loop supports storing DNSSEC private keys in HSMs and performing cryptographic operations on them using PKCS#11. Loop makes use of PKCS#11 through an OpenSSL 3 provider module called `pkcs11-provider`.

Note

- OpenSSL 3's plug-in mechanism is called an *OpenSSL provider*
- A HSM vendor-supplied library which implements the PKCS#11 API is called a *PKCS#11 provider*

These are two separate concepts.

6.12.1 Example of PKCS#11 usage

This is an example of how to use Loop's PKCS#11 support to perform DNSSEC zone signing with keys stored in HSMs (hardware security modules).

- This example uses the `SoftHSMv2` PKCS#11 provider for demonstration.

- In real-world usage, you would use the PKCS#11 provider module provided by your HSM vendor.
- For [some smart cards and USB tokens](#), you may be able to use the PKCS#11 provider provided by the [OpenSC](#) project.

To bridge between the PKCS#11 provider module and the OpenSSL library, an OpenSSL provider module called [pkcs11-provider](#) is used.

Programs in **Loop** call the **OpenSSL library**, which calls the **pkcs11-provider** OpenSSL provider, which in turn calls the HSM vendor's **PKCS#11 provider** (**SoftHSMv2** in this example).

6.12.1.1 Install dependencies

As the `root` user or using `sudo`, install the **SoftHSMv2**, **pkcs11-provider**, and **p11-kit** packages for your platform. **p11-kit** provides the program `p11tool` which you may optionally use to manipulate tokens on the HSM.

For example, on Red Hat Enterprise Linux 10 or Fedora platforms, you may use the following command to install these packages:

```
[user@host ~]$ sudo dnf install softhsm pkcs11-provider p11-
↪kit
```

On RHEL, the `softhsm` package should be available as a part of AppStream.

6.12.1.2 Create a working directory

For this example, we will create a sub-directory called `pkcs11/` as our working directory and store almost everything relative to this directory. In real-world usage, you can store configuration and data in system directories and modify paths accordingly.

```
[user@host ~]$ mkdir pkcs11
[user@host ~]$ cd pkcs11
[user@host ~/pkcs11]$
```

6.12.1.3 Configure SoftHSMv2

Let us configure and setup SoftHSMv2. SoftHSMv2 stores tokens in a directory on the filesystem. We create a sub-directory within the `pkcs11/` directory for it, and create a `softhsm2.conf` file to configure SoftHSMv2 to use this directory. We also set the `SOFTHSM2_CONF` environment variable to tell SoftHSMv2 that it should use our `softhsm2.conf` file.

```
[user@host ~/pkcs11]$ mkdir softhsm2-tokenidir
[user@host ~/pkcs11]$ cat > softhsm2.conf
directories.tokenidir=softhsm2-tokenidir
[user@host ~/pkcs11]$ export SOFTHSM2_CONF=softhsm2.conf
[user@host ~/pkcs11]$
```

Next, we initialize a SoftHSMv2 token so we may use it. We also assign a PIN of "1234" (you may use a different PIN if you like):

```
[user@host ~/pkcs11]$ softhsm2-util --init-token --free --
↳label "example" --pin 1234 --so-pin 1234
Slot 0 has a free/uninitialized token.
The token has been initialized and is reassigned to slot_
↳2076677265
[user@host ~/pkcs11]$ softhsm2-util --token example --show-
↳slots
Available slots:
Slot 2076677265
  Slot info:
    Description:      SoftHSM slot ID 0x7bc79491
    Manufacturer ID:  SoftHSM project
    Hardware version: 2.6
    Firmware version: 2.6
    Token present:    yes
  Token info:
    Manufacturer ID:  SoftHSM project
    Model:            SoftHSM v2
    Hardware version: 2.6
    Firmware version: 2.6
    Serial number:    74a2bc56fbc79491
    Initialized:      yes
    User PIN init.:   yes
    Label:            example
Slot 1
  Slot info:
    Description:      SoftHSM slot ID 0x1
    Manufacturer ID:  SoftHSM project
    Hardware version: 2.6
    Firmware version: 2.6
    Token present:    yes
  Token info:
    Manufacturer ID:  SoftHSM project
    Model:            SoftHSM v2
    Hardware version: 2.6
    Firmware version: 2.6
    Serial number:
    Initialized:      no
    User PIN init.:   no
    Label:
[user@host ~/pkcs11]$
```

(The slot number it assigns may differ for you as it is chosen randomly; it does not matter.)

We can use `p11tool` to determine the PKCS#11 URI for the token. This URI will be

used later in our example.

```
[user@host ~/pkcs11]$ p11tool --list-token-urls | grep
↪ "example"
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;
↪ serial=74a2bc56fbc79491;token=example
[user@host ~/pkcs11]$
```

You can also notice newly created files in `softhsm2-tokendir/`:

```
[user@host ~/pkcs11]$ find softhsm2-tokendir/
softhsm2-tokendir/
softhsm2-tokendir/b595ca7b-14c8-d5fd-74a2-bc56fbc79491
softhsm2-tokendir/b595ca7b-14c8-d5fd-74a2-bc56fbc79491/token.
↪ object
softhsm2-tokendir/b595ca7b-14c8-d5fd-74a2-bc56fbc79491/
↪ generation
softhsm2-tokendir/b595ca7b-14c8-d5fd-74a2-bc56fbc79491/token.
↪ lock
[user@host ~/pkcs11]$
```

6.12.1.4 Configure OpenSSL

Let us now configure **OpenSSL** to use **pkcs11-provider**, and **pkcs11-provider** to use **SoftHSMv2**. We create an `openssl.cnf` file to configure OpenSSL. We also set the `OPENSSL_CONF` environment variable to tell OpenSSL that it should use our `openssl.cnf` file.

```
[user@host ~/pkcs11]$ cat > openssl.cnf
HOME = .

openssl_conf = openssl_init

[openssl_init]
providers = provider_sect

[provider_sect]
default = default_sect
pkcs11 = pkcs11_sect

[default_sect]
activate = 1

[pkcs11_sect]
module = pkcs11.so
pkcs11-module-path = /usr/lib/softhsm/libsofthsm2.so
pkcs11-module-token-pin = 1234
pkcs11-module-quirks = no-deinit
```

(continues on next page)

(continued from previous page)

```
activate = 1
[user@host ~/pkcs11]$ export OPENSSL_CONF=openssl.cnf
[user@host ~/pkcs11]$
```

In the above configuration, a `pkcs11` OpenSSL provider module is configured in the `pkcs11_sect` section. It uses the `pkcs11.so` module installed as part of the **pkcs11-provider** package into the OpenSSL `openssl-modules` directory (typically somewhere within `/usr/lib/`).

- `pkcs11-module-path` configures **pkcs11-provider** to use the **SoftHSM2** PKCS#11 provider. In case this path is incorrect, please use the correct path to `libsofthsm2.so` on your host.
- `pkcs11-mode-token-pin` configures the PIN to access the **SoftHSMv2** token. It is configured here as it is a tutorial, but you may want to leave it out for security, and you'll be prompted to provide the PIN every time the token is used.
- `pkcs11-module-quirks` configures quirks in **pkcs11-provider**. Here, `no-deinit` is configured to ask **pkcs11-provider** not to perform deinitialization during shutdown of the PKCS#11 module. On some supported platforms, the currently available version of **SoftHSMv2** does not deinitialize properly and crashes. The available quirks are documented in the [provider-pkcs11](#) manpage.

Please consult the OpenSSL documentation for more details about what the various configuration elements mean.

6.12.1.5 Create a DNSKEY by generating or importing keys

We will create a DNSKEY to sign the `example.com.` zone. There are two approaches to creating it. We can:

- Generate a new DNSKEY on the HSM
- Import an existing key from the HSM

In both cases, the `.private` file of the DNSKEY file pair that is created contains a **label** referring to the HSM. It doesn't contain any private key material (which lives within the HSM). The `.key` file of the DNSKEY file pair that is created contains the public key as a DNSKEY record.

Both approaches are described in the section below.

6.12.1.5.1 Create a DNSKEY by generating it

The `dnssec-keygen(1)` program can be used to generate a key on a HSM. We have to provide a **label** or a **full PKCS#11** URI for an object to store the key into, using the `dnssec-keygen -l` argument. In this case, we will use a PKCS#11 URI, and label the key as `my-rsa-2048`. So we append `;object=my-rsa-2048` to the URI that `pl1tool` returned above.

We create a DNSKEY on the HSM with algorithm `RSASHA256`; the RSA key is 2048 bits in size:

```
[user@host ~/pkcs11]$ dnssec-keygen -l "pkcs11:model=SoftHSM
↳%20v2;manufacturer=SoftHSM%20project;
↳serial=74a2bc56fbc79491;token=example;object=my-rsa-2048" -
↳a RSASHA256 -b 2048 -n ZONE example.com
Generating key pair.
Kexample.com.+008+44679
[user@host ~/pkcs11]$ cat Kexample.com.+008+44679.private
Private-key-format: v1.3
Algorithm: 8 (RSASHA256)
Modulus:
↳pyBJmKobVVB4NJpEYqUIsY46DZwH47dYCDCQX4qGJ2d2wbX8h9wdw8onhSpgpF5BiqfJ0uK
↳OwsPa5FkYRRouTxXtP2BoWpq+ZRWBjssn+H7Z7Zn7BMNVyZpMiYaC5eMqWDT9N6HnD8fa6h
↳nwTI22SJdnZpwhDpanqn1Li4zgH/
↳ciy3ktdkVxD+3cgp07mSO2PowrDv7Qeyga3tRyYGO3rwHxPsbF8syj7FNCpWwFbTOUS67oo
↳ekPpUwcbuZuBiX0UAr9Q==
PublicExponent: AQAB
Label:
↳cGtjczExOmlvZGVsPVNvZnRIU00lMjB2MjttYW5lZmFjdHVyZXI9U29mdEhTTSUyMHByb2p
Created: 20250705122326
Publish: 20250705122326
Activate: 20250705122326
[user@host ~/pkcs11]$ cat Kexample.com.+008+44679.key
; This is a zone-signing key, keyid 44679, for example.com.
; Created: 20250705122326 (Sat Jul 5 20:23:26 2025)
; Publish: 20250705122326 (Sat Jul 5 20:23:26 2025)
; Activate: 20250705122326 (Sat Jul 5 20:23:26 2025)
example.com. IN DNSKEY 256 3 8
↳AwEAAacgSZiqG1VQeDSaRGKlCLGO0g2cB+O3WAgwkF+KhidndsG1/Ifc
↳HcPKJ4UqYKReQYqnydLihAcfzsLD2uRZGEUaLk8V7T9gaFqavmUVgSbL J/
↳h+2e2Z+wTDVcmaTImGguXjKlg0/Teh5w/H2uoU8fJgvub4U0++gOxR
↳OK+W7pB+PEVlHyCYeIdYqRtrBm8mv58EyNtkiXZ2acIQ6Wp6p9S4uM4B /
↳3Ist5LXZFcQ/t3IKd05kjtj6MKw7+0HsoGt7UcmBjt68B8T7GxfLMo+
↳xTQqVsBW0zlEu6KmpD8Vd0YXrzjhU49UkR+JqW7rzWuEWP3pD6VMHG7
↳mbgYl9FAK/U=
[user@host ~/pkcs11]$
```

The value of the `Label:` field printed above from the `Kexample.com.+008+44679.private` file is a Base64 encoded PKCS#11 URI string of the object. There is no private key material in this file, and this file serves like a symbolic link to the object `my-rsa-2048` in the HSM.

```
[user@host ~/pkcs11]$ echo -n
↳"cGtjczExOmlvZGVsPVNvZnRIU00lMjB2MjttYW5lZmFjdHVyZXI9U29mdEhTTSUyMHByb2p
↳" | base64 -d; echo
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;
↳serial=74a2bc56fbc79491;token=example;object=my-rsa-2048
[user@host ~/pkcs11]$
```


(The DNSKEY key ID it assigns may differ for you as it is chosen randomly; it does not matter.)

See the *dnssec-keygen(1)* manpage for more details on its usage.

6.12.1.5.2 Create a DNSKEY by importing it from the HSM

Alternatively, the *dnssec-keyfromlabel(1)* program can be used to import an existing key from a HSM. The public key is imported. We have to provide a **label** or a full PKCS#11 URI of an object to import the key from, using the *dnssec-keyfromlabel -l* argument. In this case, we will use a PKCS#11 URI, and assume the existing object is labeled *my-rsa-2048*. So we append *;object=my-rsa-2048* to the URI that *pl1tool* returned above.

We create a DNSKEY with algorithm RSASHA256 by importing an existing key labelled *my-rsa-2048* from the HSM:

```
[user@host ~/pkcs11]$ dnssec-keyfromlabel -l
↪ "pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;
↪ serial=74a2bc56fbc79491;token=example;object=my-rsa-2048" -
↪ a RSASHA256 -n ZONE example.com
Kexample.com.+008+44679
[user@host ~/pkcs11]$ cat Kexample.com.+008+44679.key
; This is a zone-signing key, keyid 44679, for example.com.
; Created: 20250705125450 (Sat Jul  5 20:54:50 2025)
; Publish: 20250705125450 (Sat Jul  5 20:54:50 2025)
; Activate: 20250705125450 (Sat Jul  5 20:54:50 2025)
example.com. IN DNSKEY 256 3 8
↪ AwEAAacgSZiqG1VQeDSaRGKlCLGO0g2cB+O3WAgwkF+KhidndsG1/Ifc
↪ HcPKJ4UqYKReQYqnydLihAcfzsLD2uRZGEUaLk8V7T9gaFqavmUVgSbL J/
↪ h+2e2Z+wTDVcmaTImGguXjKlg0/Teh5w/H2uoU8fJgvub4U0++gOxR
↪ OK+W7pB+PEV1HyCYeIdYqRtrBm8mv58EyNtkiXZ2acIQ6Wp6p9S4uM4B /
↪ 3Ist5LXZFcQ/t3IKdO5kjtj6MKw7+0HsoGt7UcmBjt68B8T7GxfLMo+
↪ xTQqVsBW0zlEu6KMpD8Vd0YXrzjhU49UkR+JqW7rzWuEWP3pD6VMHG7
↪ mbgYl9FAK/U=
[user@host ~/pkcs11]$ cat Kexample.com.+008+44679.private
Private-key-format: v1.3
Algorithm: 8 (RSASHA256)
Modulus:
↪ pyBJmKobVVB4NJpEYqUIsY46DZwH47dYCDCQX4qGJ2d2wbX8h9wdw8onhSpgpF5BiqfJ0uK
↪ OwsPa5FkYRRouTxXtP2BoWpq+ZRWBjssn+H7Z7Zn7BMNVyZpMiYaC5eMqWDT9N6HnD8fa6h
↪ nwTI22SJdnZpwhDpanqn1Li4zgH/
↪ ciy3ktdkVxD+3cgp07mSO2PowrDv7Qeyga3tRyYGO3rwHxPsbF8syj7FNCpWwFbTOUS67oo
↪ ekPpUwcbuZuBiX0UAr9Q==
PublicExponent: AQAB
Label:
↪ cGtjczExOmlvZGVsPVNVZnRIU00lMjB2MjttYW5lZmFjdHVyZXI9U29mdEhTTSUyMHByb2p
Created: 20250705125450
```

(continues on next page)

(continued from previous page)

```
Publish: 20250705125450
Activate: 20250705125450
[user@host ~/pkcs11]$
```

Unlike the `dnssec-keygen` command in the previous section, no key size can be specified when using `dnssec-keyfromlabel` as the key is already present in the HSM and has its existing key size.

The value of the `Label:` field printed above from the `Kexample.com.+008+44679.private` file is a Base64 encoded PKCS#11 URI string of the object. There is no private key material in this file, and this file serves like a symbolic link to the object `my-rsa-2048` in the HSM.

```
[user@host ~/pkcs11]$ echo -n
↪ "cGtjczExOmlvZGVsPVNvZnRIU00lMjB2MjttYW51ZmFjdHVyZXI9U29mdEhTTSUyMHByb2
↪ " | base64 -d; echo
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;
↪ serial=74a2bc56fbc79491;token=example;object=my-rsa-2048
[user@host ~/pkcs11]$
```

See the `dnssec-keyfromlabel (1)` manpage for more details on its usage.

6.12.1.6 Sign a DNS zone using the DNSKEY

Now that we have the `Kexample.com.+008+44679.key` and `Kexample.com.+008+44679.private` files, we can sign a master file for the `example.com` zone. We use the `dnssec-signzone (1)` program to sign zones.

Let's create a sample master file for the `example.com.` zone, and also check that it is syntactically correct:

```
[user@host ~/pkcs11]$ cat > example.com.db
$TTL 3600
example.com.      IN SOA  . . 2025070500 600 600 1200 600
example.com.      NS ns1.example.com.
ns1.example.com.  A 127.0.0.1
example.com.      A 127.0.0.1
[user@host ~/pkcs11]$ named-checkzone example.com example.com.
↪ db
zone example.com/IN: loaded serial 2025070500
OK
[user@host ~/pkcs11]$
```

This zone's data can be signed now using the HSM. We use smart signing by calling `dnssec-signzone -S` so that it discovers the DNSKEY automatically in the `Kexample.com.+008+44679.key` and `Kexample.com.+008+44679.private` files.

```
[user@host ~/pkcs11]$ dnssec-signzone -S -z -o example.com.
↪example.com.db
Fetching ZSK 44679/RSASHA256 from key repository.
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 0 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked
example.com.db.signed
[user@host ~/pkcs11]$
```

The signed zone's data is saved to the master file `example.com.db.signed` and has the following content:

```
[user@host ~/pkcs11]$ cat example.com.db.signed
; File written on Sat Jul  5 21:22:21 2025
; dnssec-signzone version 1.99.3.20250623003957.08c26b66c1
example.com.      3600      IN SOA  . . (
                    2025070500 ; serial
                    600        ; refresh (10 minutes)
                    600        ; retry  (10 minutes)
                    1200       ; expire  (20 minutes)
                    600        ; minimum (10 minutes)
                    )
                3600      RRSIG   SOA 8 2 3600 (
                    20250804122221 20250705122221 44679 ↪
↪example.com.

                    gOVYSFEU+F6oApb4xEFAhj77VI8XGhVxwiU+
                    LzVySfD/j6yL1kCGOez/f/c4jxxZzCP36ztB
                    MLejT6A/fuSfGVIRRsjkQ+2AK+p3z0wiSyYL
                    zSMedgiQqjh5JwQ+CaR/lAa2+AzQG6goe76k
                    3ZQbda6E/4EK733jA2Ahh3Af15YUJH3yqvDo
                    ZtVavZxd+Jo1c7FQBKn24NotXDjqewaCZ3XX
                    TxmmFSBzdy1Vv3PpnapF4FzLvnWqsTbR6v/Z
                    JQPqKvxGjUDc5ETp+aOG59JkzSE9vPj/bGZQ
                    OiQhUQ0VL4iDpyiN98+y8oQ3Tvp9Lh7KNwRW
                    beC9Lgf8CFxNg5BgUw== )
                3600      NS   ns1.example.com.
                3600      RRSIG   NS 8 2 3600 (
                    20250804122221 20250705122221 44679 ↪
↪example.com.

                    IdGBtB2RUyTEcpxxCoVqa6XKyYaMgyGaZhp3
                    dj31FNV7xyH2xLM7j9d5A8qr1ORMLm31UCB/
                    0tLcxndqFcH0gXipxm78rppEFGiLSNPEDs1q
                    8owr+zzjJ2Dtkf5Sg008BmDrAMFMH1vHQ2E0
                    Ij8wjdrZ11NDVkdwkW/MPMp1B1RSTLZujXx3
                    2LQXMs1RQcrc4Nk70juyl8UNe2nUOJuEO6SV
                    2Y8YtDqCdWa2bgmrncm7OE+SJ79UZOG6gsu/
```

(continues on next page)

(continued from previous page)

```

3600 688V39AaN/IMK57Ua59aeN6C7bZQ3D8YWKo5
3600 IDS+Mza+jAeydQpjFEZSBC5Z9sGREIpPRyPg
3600 3euahyjMhnX5DWkZUg== )
3600 A 127.0.0.1
3600 RRSIG A 8 2 3600 (
3600 20250804122221 20250705122221 44679
example.com.
mJ1Ma0CgANegpwh40x6wIxxMiO/W99PLE66w
mteJCTiJWdoNjiQsaK8CPpaehP+ttF4WazDs
0DNaZUZRBiaWtLa5PIJGRYlfHB/gWkWRy3Oq
3HeaVRU0i59GW0/cCt4oJsJF3KPziFLIdmUU
Ge9GA7x17m+Fyp/BDo6EfPsYgOZlMY+GBofC
DfxsZzgPhI6+MN6uLkDLrmwtuZ9KS3liu4I5
tdIx9PLRteH1CEEAhUAHq0wFAs39TQs2uB27
TiHzqTjfergtPyZYHWc7Swry3JahKLcQw0kB
n+4q1tbb8ktOZqUszrFMt+xKzi948rkFJswU
DFID+NQzpSNRGtOzaw== )
600 NSEC ns1.example.com. A NS SOA RRSIG NSEC
DNSKEY
600 RRSIG NSEC 8 2 600 (
600 20250804122221 20250705122221 44679
example.com.
kYFu3pdpBfI+7CITOXdKULrwCpbmVQlO+jC/
sNPcdDSNuInm8cq5UVy05/vV0FxPSJEhY1qr
38FQbYl9CAI0EuIDvlwhadlO6kli4A4QVlER
TB6diBOOHV00eyUYOwg/gNYDKaKGdh1msAhV
QnvIScf9Z5QgCglnA+vOuVngGM8u4jNQjUTf
vReOgqr4ZoWfFommJHVMzbtM8gHVabPw4ExB
hoAeqe4o4ZPPReBWh75ounOUXMhrIS48YJ5/
/X6K54m7tAKEHL7yQ21sM5OELNWe+CqXnYcK
Bq6NjYPRr+aNwqpIQ+SmsIGhE9CBbrwgka39
JpLv3QxzY+CUNaJJmA== )
3600 DNSKEY 256 3 8 (
3600 AwEAAacgSZiqG1VQeDSaRGKlCLGOOg2cB+O3
3600 WAgwkF+KhidndsG1/IfcHcPKJ4UqYKReQYqn
3600 ydLihAcfzsLD2uRZGEUaLk8V7T9gaFqavmUV
3600 gSbLJ/h+2e2Z+wTDVcmaTImGguXjKlg0/Teh
3600 5w/H2uoU8fJgvub4U0++gOxROK+W7pB+PEV1
3600 HyCYeIdYqRtrBm8mv58EyNtkiXZ2acIQ6Wp6
3600 p9S4uM4B/3Ist5LXZFcQ/t3IKdO5kjtj6MKw
3600 7+0HsoGt7UcmBjt68B8T7GxfLMo+xTQqVsBW
3600 0zlEuu6KMpD8Vd0YXrzjhU49UkR+JqW7rzWu
3600 EWP3pD6VMHG7mbgYl9FAK/U=
3600 ) ; ZSK; alg = RSASHA256 ; key id = 44679
3600 RRSIG DNSKEY 8 2 3600 (
3600 20250804122221 20250705122221 44679

```

(continues on next page)

(continued from previous page)

```

↪example.com.
GRds9ebVM8g6n6GqswPoiaEV5xK+2DcxSXZG
qgLWu8eSakGAoZQkjdz983NhL1cGDfk5q7pL
dvxpnCNZ4tzg7AKKfXr2BEqIGt7Xo9KLtKt0
SdsF5RioWyL3iRGjeFbaqxK5G2Dy1CCQD7DJ
KHvmv+pfnplls3i6xKNRjXsVrNPF1iEyivt8
l5NF1AsaHOfk/PIMrlpihzFBxM2A9S8XXwn7
j6KYm15FWl9RmU2y5akzPTntVJPFVbVJ0OcD
DNIuNd4kzF8IWY3zigrfhuH3xPzGAhop30PH
r9ANnTASD7gzcVPGjkyfSXaxOBVKTtK/cmpP
/IK+CLaMliL3fOfHyA== )
ns1.example.com. 3600 IN A 127.0.0.1
3600 RRSIG A 8 3 3600 (
20250804122221 20250705122221 44679 ↵
↪example.com.
jTuSieMy/0Faveu9ODXYAZ8NTIZqo1zJ//IC
syNX3Fs908G+sUqyZNnH0JHEC5cJQvfj0OEK
Mx8aW7PkyO7gKSraUDHs5F8fws0WNNueY+Oe
Zi6H6NA1l99wUxKLcaakzFZj6XgKYCLwSO1J
yyQrPb9Zs6FNkx+GaaeRbKq7xHeK/bMTMggt
7P3hhsFfYIj4KI0hgP23Jomc+OePZe6eDHee
WUHpod5TnLDXOri7IkLLuHxUsLo9Xnjp9FF5
HJcCvx1/XRip6D1kJqZ0x6w1kTEaA0E5Yff0
6uVfxB5kjmiN40EoFdpvfylRDZVQPPey214e
MTkvp3IOwDgqb8yMrQ== )
600 NSEC example.com. A RRSIG NSEC
600 RRSIG NSEC 8 3 600 (
20250804122221 20250705122221 44679 ↵
↪example.com.
KmGYCG01vI2SFsbP6NhXTyDWEzr08MF7Qiy9
TmqK5gL4cTwwIMhsAUWJG3YXMKGiotpMBsW5
kvso3od/+fisEGrDA9ik4higHwTZY+aocX1O
DLUTqNaVQ/KJ/vH096jtFb8lkvyfuFO+K/tY
GXj+sd6Ol3sdRyUkAU17GRpchLgAjFGWImX8
s5WXfWDpmCVMSy7ORQxX6qfUhuqSIGcKjGbi
o1Uw1n4D7LxRSqbKgLD5i+YVPyUjOipdTzZm
c+X6qfUF5lflbXuJbto8MSORwUGyA9jbG64H
mV8xPAV3lI627RNCOretwtAl2sRMr2c15XAY
M8NLrOtLjl6xBCKe4A== )
[user@host ~/pkcs11]$

```

See the *dnssec-signzone(1)* manpage for more details on its usage.

6.12.1.7 Verify the signed zone's data

Though the signed zone's data is automatically verified as part of the signing process, we can explicitly verify the signed zone's data to check that it is signed properly using the `dnssec-verify(1)` program:

```
[user@host ~/pkcs11]$ dnssec-verify -z -o example.com example.
com.db.signed
Loading zone 'example.com' from file 'example.com.db.signed'
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 0 active, 0 stand-by, 0 revoked
                  ZSKs: 1 active, 0 stand-by, 0 revoked
[user@host ~/pkcs11]$
```

See the `dnssec-verify(1)` manpage for more details on its usage.

This ends the example.

6.13 IPv6

Loop fully supports all currently defined forms of IPv6 name to address and address to name lookups. It will also use IPv6 addresses to make queries when running on an IPv6 capable system.

For forward lookups, Loop supports only AAAA records. RFC 3363 deprecated the use of A6 records, and client-side support for A6 records is not present in Loop. However, authoritative Loop name servers still load zone files containing A6 records correctly, answer queries for A6 records, and accept zone transfer for a zone containing A6 records.

For IPv6 reverse lookups, Loop supports the traditional "nibble" format used in the `ip6.arpa` domain, as well as the older, deprecated `ip6.int` domain. Per RFC 3363, there is no support for "binary labels" (also known as "bitstring") format. In particular, an authoritative Loop name server will not load a zone file containing binary labels.

For an overview of the format and structure of IPv6 addresses, see *section_title*.

6.13.1 Address Lookups Using AAAA Records

The IPv6 AAAA record is a parallel to the IPv4 A record, and, unlike the deprecated A6 record, specifies the entire IPv6 address in a single record. For example,

```
$ORIGIN example.com.
host          3600      IN      AAAA    2001:db8::1
```

Use of IPv4-in-IPv6 mapped addresses is not recommended. If a host has an IPv4 address, use an A record, not a AAAA, with `::ffff:192.168.42.1` as the address.

6.13.2 Address to Name Lookups Using Nibble Format

When looking up an address in nibble format, the address components are simply reversed, just as in IPv4, and `ip6.arpa.` is appended to the resulting name. For example, the following would provide reverse name lookup for a host with address `2001:db8::1`.

```
$ORIGIN 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.  
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 14400 IN PTR (  
                                host.example.com. )
```

6.14 Empty zones

named has some built-in "empty" zones containing SOA and NS records only. These are for zones that should normally be answered locally and which queries should not be sent to the internet. The official servers which cover these namespaces return NX-DOMAIN responses to these queries. In particular, these cover the reverse namespaces for addresses from [RFC 1918](#), [RFC 3849](#), [RFC 4193](#), [RFC 5735](#), [RFC 5737](#), [RFC 6598](#), [RFC 7534](#), and [RFC 8375](#). They also include the reverse namespace for IPv6 local address (locally assigned), IPv6 link local addresses, the IPv6 loopback address, and the IPv6 unknown address. The [IANA IPv4 Special-Purpose Address Registry](#) and the [IANA IPv6 Special-Purpose Address Registry](#) now track all special-purpose address blocks.

named will attempt to determine if a built-in zone already exists or is active (covered by a *forward-only* forwarding declaration) and will not create an empty zone in that case.

The current list of empty zones is:

- 10.IN-ADDR.ARPA
- 16.172.IN-ADDR.ARPA
- 17.172.IN-ADDR.ARPA
- 18.172.IN-ADDR.ARPA
- 19.172.IN-ADDR.ARPA
- 20.172.IN-ADDR.ARPA
- 21.172.IN-ADDR.ARPA
- 22.172.IN-ADDR.ARPA
- 23.172.IN-ADDR.ARPA
- 24.172.IN-ADDR.ARPA
- 25.172.IN-ADDR.ARPA
- 26.172.IN-ADDR.ARPA
- 27.172.IN-ADDR.ARPA

- 28.172.IN-ADDR.ARPA
- 29.172.IN-ADDR.ARPA
- 30.172.IN-ADDR.ARPA
- 31.172.IN-ADDR.ARPA
- 168.192.IN-ADDR.ARPA
- 64.100.IN-ADDR.ARPA
- 65.100.IN-ADDR.ARPA
- 66.100.IN-ADDR.ARPA
- 67.100.IN-ADDR.ARPA
- 68.100.IN-ADDR.ARPA
- 69.100.IN-ADDR.ARPA
- 70.100.IN-ADDR.ARPA
- 71.100.IN-ADDR.ARPA
- 72.100.IN-ADDR.ARPA
- 73.100.IN-ADDR.ARPA
- 74.100.IN-ADDR.ARPA
- 75.100.IN-ADDR.ARPA
- 76.100.IN-ADDR.ARPA
- 77.100.IN-ADDR.ARPA
- 78.100.IN-ADDR.ARPA
- 79.100.IN-ADDR.ARPA
- 80.100.IN-ADDR.ARPA
- 81.100.IN-ADDR.ARPA
- 82.100.IN-ADDR.ARPA
- 83.100.IN-ADDR.ARPA
- 84.100.IN-ADDR.ARPA
- 85.100.IN-ADDR.ARPA
- 86.100.IN-ADDR.ARPA
- 87.100.IN-ADDR.ARPA
- 88.100.IN-ADDR.ARPA
- 89.100.IN-ADDR.ARPA
- 90.100.IN-ADDR.ARPA

- 91.100.IN-ADDR.ARPA
- 92.100.IN-ADDR.ARPA
- 93.100.IN-ADDR.ARPA
- 94.100.IN-ADDR.ARPA
- 95.100.IN-ADDR.ARPA
- 96.100.IN-ADDR.ARPA
- 97.100.IN-ADDR.ARPA
- 98.100.IN-ADDR.ARPA
- 99.100.IN-ADDR.ARPA
- 100.100.IN-ADDR.ARPA
- 101.100.IN-ADDR.ARPA
- 102.100.IN-ADDR.ARPA
- 103.100.IN-ADDR.ARPA
- 104.100.IN-ADDR.ARPA
- 105.100.IN-ADDR.ARPA
- 106.100.IN-ADDR.ARPA
- 107.100.IN-ADDR.ARPA
- 108.100.IN-ADDR.ARPA
- 109.100.IN-ADDR.ARPA
- 110.100.IN-ADDR.ARPA
- 111.100.IN-ADDR.ARPA
- 112.100.IN-ADDR.ARPA
- 113.100.IN-ADDR.ARPA
- 114.100.IN-ADDR.ARPA
- 115.100.IN-ADDR.ARPA
- 116.100.IN-ADDR.ARPA
- 117.100.IN-ADDR.ARPA
- 118.100.IN-ADDR.ARPA
- 119.100.IN-ADDR.ARPA
- 120.100.IN-ADDR.ARPA
- 121.100.IN-ADDR.ARPA
- 122.100.IN-ADDR.ARPA

- You can configure empty zones by using the `empty-server`, `empty-contact`, `empty-zones-enable`, and the `disable-empty-zone` options of `named.conf(5)`. Empty zones can be set at the view-level and only apply to views of class `IN`. Disabled empty zones are only inherited from the `options` statement if there are no disabled empty zones specified at the view-level. To override the list of disabled zones from the `options` statement, you can disable the root zone at the view-level. For example:

If you are using the address ranges covered here, you should already have reverse zones covering the addresses you use. In practice this appears to not be the case with many queries being made to the infrastructure servers for names in these spaces. So many in fact that sacrificial servers were needed to be deployed to channel the query load away from the infrastructure servers.

Note

The real parent servers for these zones should disable all empty zone under the parent zone they serve. For the real root servers, this is all built-in empty zones. This will enable them to return referrals to deeper in the tree.

6.15 DNS64

```
acl rfc1918 { 10/8; 192.168/16; 172.16/12; };

dns64 64:FF9B::/96 {
    clients { any; };
    mapped { !rfc1918; any; };
    exclude { 64:FF9B::/96; ::ffff:0000:0000/96; };
    suffix ::;
};
```

6.16 Forwarding queries

The forwarding facility can be used to create a large site-wide cache on a few servers, reducing traffic over links to external name servers. It can also be used to allow queries by servers that do not have direct access to the Internet, but wish to look up exterior names anyway. Forwarding occurs only on those queries for which the server is not authoritative and does not have the answer in its cache.

forward

forwarders

Forwarding can also be configured on a per-domain basis, allowing for the global forwarding options to be overridden in a variety of ways. You can set particular domains to use different forwarders, or have a different `forward only/first` behavior, or not forward at all, see *section_title*.

6.17 Dual-stack Servers

Dual-stack servers are used as servers of last resort to work around problems in reachability due the lack of support for either IPv4 or IPv6 on the host machine.

dual-stack-servers

6.18 Access Control

Access to the server can be restricted based on the IP address of the requesting system. See *section_title* for details on how to specify IP address lists.

allow-notify
allow-query
allow-query-on
allow-query-cache
allow-query-cache-on
allow-recursion
allow-recursion-on
allow-update
allow-update-forwarding
allow-transfer
block
no-case-compress
resolver-query-timeout

6.19 Interfaces

The interfaces and ports that the server will answer queries from may be specified using the `listen-on` option. `listen-on` takes an optional port and an `address_match_list` of IPv4 addresses. (IPv6 addresses are ignored, with a logged warning.) The server will listen on all interfaces allowed by the address match list. If a port is not specified, port 53 will be used.

Multiple `listen-on` statements are allowed. For example,

```
listen-on { 5.6.7.8; };  
listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

will enable the name server on port 53 for the IP address 5.6.7.8, and on port 1234 of an address on the machine in net 1.2 that is not 1.2.3.4.

If no `listen-on` is specified, the server will listen on port 53 on all IPv4 interfaces.

The `listen-on-v6` option is used to specify the interfaces and the ports on which the server will listen for incoming queries sent using IPv6. If not specified, the server will listen on port 53 on all IPv6 interfaces.

When

```
{ any; }
```

is specified as the `address_match_list` for the `listen-on-v6` option, the server does not bind a separate socket to each IPv6 interface address as it does for IPv4 if the operating system has enough API support for IPv6 (specifically if it conforms to RFC 3493 and RFC 3542). Instead, it listens on the IPv6 wildcard address. If the system

only has incomplete API support for IPv6, however, the behavior is the same as that for IPv4.

A list of particular IPv6 addresses can also be specified, in which case the server listens on a separate socket for each specified address, regardless of whether the desired API is supported by the system. IPv4 addresses specified in `listen-on-v6` will be ignored, with a logged warning.

Multiple `listen-on-v6` options can be used. For example,

```
listen-on-v6 { any; };  
listen-on-v6 port 1234 { !2001:db8::/32; any; };
```

will enable the name server on port 53 for any IPv6 addresses (with a single wildcard socket), and on port 1234 of IPv6 addresses that is not in the prefix 2001:db8::/32 (with separate sockets for each matched address.)

To make the server not listen on any IPv6 address, use

```
listen-on-v6 { none; };
```

6.20 Query address

If the server doesn't know the answer to a question, it will query other name servers. `query-source` specifies the address and port used for such queries. For queries sent over IPv6, there is a separate `query-source-v6` option. If address is `*` (asterisk) or is omitted, a wildcard IP address (`INADDR_ANY`) will be used.

If port is `*` or is omitted, a random port number from a pre-configured range is picked up and will be used for each query. The port range(s) is that specified in the `use-v4-udp-ports` (for IPv4) and `use-v6-udp-ports` (for IPv6) options, excluding the ranges specified in the `avoid-v4-udp-ports` and `avoid-v6-udp-ports` options, respectively.

The defaults of the `query-source` and `query-source-v6` options are:

```
query-source address * port *;  
query-source-v6 address * port *;
```

If `use-v4-udp-ports` or `use-v6-udp-ports` is unspecified, `named` will check if the operating system provides a programming interface to retrieve the system's default range for ephemeral ports. If such an interface is available, `named` will use the corresponding system default range; otherwise, it will use its own defaults:

```
use-v4-udp-ports { range 1024 65535; };  
use-v6-udp-ports { range 1024 65535; };
```

Note: make sure the ranges be sufficiently large for security. A desirable size depends on various parameters, but we generally recommend it contain at least 16384 ports (14 bits of entropy). Note also that the system's default range when used may be too small

for this purpose, and that the range may even be changed while `named` is running; the new range will automatically be applied when `named` is reloaded. It is encouraged to configure `use-v4-udp-ports` and `use-v6-udp-ports` explicitly so that the ranges are sufficiently large and are reasonably independent from the ranges used by other applications.

Note: the operational configuration where `named` runs may prohibit the use of some ports. For example, UNIX systems will not allow `named` running without a root privilege to use ports less than 1024. If such ports are included in the specified (or detected) set of query ports, the corresponding query attempts will fail, resulting in resolution failures or delay. It is therefore important to configure the set of ports that can be safely used in the expected operational environment.

The defaults of the `avoid-v4-udp-ports` and `avoid-v6-udp-ports` options are:

```
avoid-v4-udp-ports {};  
avoid-v6-udp-ports {};
```

Note: It is generally strongly discouraged to specify a particular port for the `query-source` or `query-source-v6` options; it implicitly disables the use of randomized port numbers.

Note

The address specified in the `query-source` option is used for both UDP and TCP queries, but the port applies only to UDP queries. TCP queries always use a random unprivileged port.

Note

See also `transfer-source` and `notify-source`.

6.21 Zone Transfers

Loop has mechanisms in place to facilitate zone transfers and set limits on the amount of load that transfers place on the system. The following options apply to zone transfers.

```
also-notify  
max-transfer-time-in  
max-transfer-idle-in  
max-transfer-time-out  
max-transfer-idle-out  
serial-query-rate  
transfers-in  
transfers-out  
transfers-per-ns
```

```
transfer-source
transfer-source-v6
alt-transfer-source
alt-transfer-source-v6
use-alt-transfer-source
notify-source
notify-source-v6
```

6.22 UDP port lists

`use-v4-udp-ports`, `avoid-v4-udp-ports`, `use-v6-udp-ports`, and `avoid-v6-udp-ports` specify a list of IPv4 and IPv6 UDP ports that will be used or not used as source ports for UDP messages. See [Query address](#) about how the available ports are determined.

For example, with the following configuration:

```
use-v6-udp-ports { range 32768 65535; };
avoid-v6-udp-ports { 40000; range 50000 60000; };
```

UDP ports of IPv6 messages sent from **named** will be in one of the ranges 32768 to 39999, 40001 to 49999, and 60001 to 65535.

`avoid-v4-udp-ports` and `avoid-v6-udp-ports` can be used to prevent **named** from choosing as its random source port a port that is blocked by your firewall or a port that is used by other applications; if a query went out with a source port blocked by a firewall, the answer would not get by the firewall and the name server would have to query again.

Note

The desired range can also be represented only with `use-v4-udp-ports` and `use-v6-udp-ports`, and the `avoid-` options are redundant in that sense; they are provided for backward compatibility and to possibly simplify the port specification.

Error

TODO: Remove the `avoid-` options.

6.23 Operating System Resource Limits

The server's usage of many system resources can be limited. Scaled values are allowed when specifying resource limits. For example, 1G can be used instead of 1073741824 to specify a limit of one gigabyte. `unlimited` requests unlimited use, or the maximum available amount. `default` uses the limit that was in force when the server was started. See the description of `size_spec` in *section_title*.

The following options set operating system resource limits for the name server process. Some operating systems don't support some or any of the limits. On such systems, a warning will be issued if the unsupported limit is used.

```
coresize
datasize
files
stacksize
```

6.24 Server Resource Limits

The following options set limits on the server's resource consumption that are enforced internally by the server rather than the operating system.

```
max-journal-size
max-records
recursive-clients
tcp-clients
clients-per-query;max-clients-per-query
fetches-per-zone
fetches-per-server
fetch-quota-params
reserved-sockets
max-cache-size
tcp-listen-queue
```

6.25 Periodic Task Intervals

```
heartbeat-interval
interface-interval
```

6.26 RRset ordering

When multiple records are returned in an answer, it may be useful to configure the order of the records placed into the response.

`rrset-order`

6.27 Tuning

`max-ncache-ttl`

`max-cache-ttl`

`sig-validity-interval`

`sig-signing-nodes`

`sig-signing-signatures`

`sig-signing-type`

`min-refresh-time;max-refresh-time;min-retry-time;max-retry-time`

`edns-udp-size`

`max-udp-size`

`max-recursion-depth`

`max-recursion-queries`

`notify-delay`

`max-rsa-exponent-size`

`prefetch`

6.28 Built-in server information zones

The server provides some helpful diagnostic information through a number of built-in zones under the pseudo-top-level-domain `loop` in the `CHAOS` class. These zones are part of a built-in view (see *section_title*) of class `CHAOS` which is separate from the default view of class `IN`. Most global configuration options (`allow-query`, etc) will apply to this view, but some are locally overridden: `notify`, `recursion` and `allow-new-zones` are always set to `no`, and `rate-limit` is set to allow three responses per second.

If you need to disable these zones, use the options below, or hide the built-in `CHAOS` view by defining an explicit view of class `CHAOS` that matches all clients.

`version`

`server-id`

6.29 Built-in Empty Zones

`empty-server`

`empty-contact`

`empty-zones-enable`

`disable-empty-zone`

6.30 Content Filtering

Loop provides the ability to filter out DNS responses from external DNS servers containing certain types of data in the answer section. Specifically, it can reject address (A or AAAA) records if the corresponding IPv4 or IPv6 addresses match the given `address_match_list` of the `deny-answer-addresses` option. It can also reject CNAME or DNAME records if the "alias" name (i.e., the CNAME alias or the substituted query name due to DNAME) matches the given `namelist` of the `deny-answer-aliases` option, where "match" means the alias name is a subdomain of one of the `name_list` elements. If the optional `namelist` is specified with `except-from`, records whose query name matches the list will be accepted regardless of the filter setting. Likewise, if the alias name is a subdomain of the corresponding zone, the `deny-answer-aliases` filter will not apply; for example, even if "example.com" is specified for `deny-answer-aliases`,

```
www.example.com. CNAME xxx.example.com.
```

returned by an "example.com" server will be accepted.

In the `address_match_list` of the `deny-answer-addresses` option, only `ip_addr` and `ip_prefix` are meaningful; any `key_id` will be silently ignored.

If a response message is rejected due to the filtering, the entire message is discarded without being cached, and a SERVFAIL error will be returned to the client.

This filtering is intended to prevent "DNS rebinding attacks," in which an attacker, in response to a query for a domain name the attacker controls, returns an IP address within your own network or an alias name within your own domain. A naive web browser or script could then serve as an unintended proxy, allowing the attacker to get access to an internal node of your local network that couldn't be externally accessed otherwise. See the paper available at <http://portal.acm.org/citation.cfm?id=1315245.1315298> for more details about the attacks.

For example, if you own a domain named "example.net" and your internal network uses an IPv4 prefix 192.0.2.0/24, you might specify the following rules:

```
deny-answer-addresses { 192.0.2.0/24; } except-from {
  ↪ "example.net"; };
deny-answer-aliases { "example.net"; };
```

If an external attacker lets a web browser in your local network look up an IPv4 address of "attacker.example.com", the attacker's DNS server would return a response like this:

```
attacker.example.com. A 192.0.2.1
```

in the answer section. Since the rdata of this record (the IPv4 address) matches the specified prefix 192.0.2.0/24, this response will be ignored.

On the other hand, if the browser looks up a legitimate internal web server "www.example.net" and the following response is returned to the Loop server

```
www.example.net. A 192.0.2.2
```

it will be accepted since the owner name "www.example.net" matches the `except-from` element, "example.net".

Note that this is not really an attack on the DNS per se. In fact, there is nothing wrong for an "external" name to be mapped to your "internal" IP address or domain name from the DNS point of view. It might actually be provided for a legitimate purpose, such as for debugging. As long as the mapping is provided by the correct owner, it is not possible or does not make sense to detect whether the intent of the mapping is legitimate or not within the DNS. The "rebinding" attack must primarily be protected at the application that uses the DNS. For a large site, however, it may be difficult to protect all possible applications at once. This filtering feature is provided only to help such an operational environment; it is generally discouraged to turn it on unless you are very sure you have no other choice and the attack is a real threat for your applications.

Care should be particularly taken if you want to use this option for addresses within 127.0.0.0/8. These addresses are obviously "internal", but many applications conventionally rely on a DNS mapping from some name to such an address. Filtering out DNS records containing this address spuriously can break such applications.

6.31 Response Policy Zones (RPZ)

Loop includes a limited mechanism to modify DNS responses for requests analogous to email anti-spam DNS blacklists. Responses can be changed to deny the existence of domains (NXDOMAIN), deny the existence of IP addresses for domains (NODATA), or contain other IP addresses or data.

Response policy zones are named in the `response-policy` option for the view or among the global options if there is no `response-policy` option for the view. Response policy zones are ordinary DNS zones containing RRsets that can be queried normally if allowed. It is usually best to restrict those queries with something like `allow-query { localhost; };`.

A `response-policy` option can support multiple policy zones. To maximize performance, a radix tree is used to quickly identify response policy zones containing triggers that match the current query. This imposes an upper limit of 32 on the number of

policy zones in a single `response-policy` option; more than that is a configuration error.

Five policy triggers can be encoded in RPZ records.

RPZ-CLIENT-IP

IP records are triggered by the IP address of the DNS client. Client IP address triggers are encoded in records that have owner names that are subdomains of `rpz-client-ip` relativized to the policy zone origin name and encode an address or address block. IPv4 addresses are represented as `prefixlength.B4.B3.B2.B1.rpz-client-ip`. The IPv4 prefix length must be between 1 and 32. All four bytes, B4, B3, B2, and B1, must be present. B4 is the decimal value of the least significant byte of the IPv4 address as in IN-ADDR.ARPA.

IPv6 addresses are encoded in a format similar to the standard IPv6 text representation, `prefixlength.W8.W7.W6.W5.W4.W3.W2.W1.rpz-client-ip`. Each of W8,...,W1 is a one to four digit hexadecimal number representing 16 bits of the IPv6 address as in the standard text representation of IPv6 addresses, but reversed as in IP6.ARPA. (Note that this representation of IPv6 address is different from IP6.ARPA where each hex digit occupies a label.) All 8 words must be present except when one set of consecutive zero words is replaced with `.zz.` analogous to double colons (`::`) in standard IPv6 text encodings. The IPv6 prefix length must be between 1 and 128.

QNAME

QNAME policy records are triggered by query names of requests and targets of CNAME records resolved to generate the response. The owner name of a QNAME policy record is the query name relativized to the policy zone.

RPZ-IP

IP triggers are IP addresses in an A or AAAA record in the ANSWER section of a response. They are encoded like client-IP triggers except as subdomains of `rpz-ip`.

RPZ-NSDNAME

NSDNAME triggers match names of authoritative servers for the query name, a parent of the query name, a CNAME for query name, or a parent of a CNAME. They are encoded as subdomains of `rpz-nsdname` relativized to the RPZ origin name. NSIP triggers match IP addresses in A and AAAA RRsets for domains that can be checked against NSDNAME policy records.

RPZ-NSIP

NSIP triggers are encoded like IP triggers except as subdomains of `rpz-nsip`. NSDNAME and NSIP triggers are checked only for names with at least `min-ns-dots` dots. The default value of `min-ns-dots` is 1 to exclude top level domains.

The query response is checked against all response policy zones, so two or more policy records can be triggered by a response. Because DNS responses are rewritten according to at most one policy record, a single record encoding an action (other than DISABLED actions) must be chosen. Triggers or the records that encode them are chosen for the rewriting in the following order:

1. Choose the triggered record in the zone that appears first in the response-policy option.
2. Prefer CLIENT-IP to QNAME to IP to NSDNAME to NSIP triggers in a single zone.
3. Among NSDNAME triggers, prefer the trigger that matches the smallest name under the DNSSEC ordering.
4. Among IP or NSIP triggers, prefer the trigger with the longest prefix.
5. Among triggers with the same prefix length, prefer the IP or NSIP trigger that matches the smallest IP address.

When the processing of a response is restarted to resolve DNAME or CNAME records and a policy record set has not been triggered, all response policy zones are again consulted for the DNAME or CNAME names and addresses.

RPZ record sets are any types of DNS record except DNAME or DNSSEC that encode actions or responses to individual queries. Any of the policies can be used with any of the triggers. For example, while the `TCP-only` policy is commonly used with `client-IP` triggers, it can be used with any type of trigger to force the use of TCP for responses with owner names in a zone.

PASSTHRU

The whitelist policy is specified by a CNAME whose target is `rpz-passthru`. It causes the response to not be rewritten and is most often used to "poke holes" in policies for CIDR blocks.

DROP

The blacklist policy is specified by a CNAME whose target is `rpz-drop`. It causes the response to be discarded. Nothing is sent to the DNS client.

TCP-Only

The "slip" policy is specified by a CNAME whose target is `rpz-tcp-only`. It changes UDP responses to short, truncated DNS responses that require the DNS client to try again with TCP. It is used to mitigate distributed DNS reflection attacks.

NXDOMAIN

The domain undefined response is encoded by a CNAME whose target is the root domain (`.`)

NODATA

The empty set of resource records is specified by CNAME whose target is the wildcard top-level domain (`*.`). It rewrites the response to NODATA or `ANCOUNT=1`.

Local Data

A set of ordinary DNS records can be used to answer queries. Queries for record types not the set are answered with NODATA.

A special form of local data is a CNAME whose target is a wildcard such as `*.example.com`. It is used as if were an ordinary CNAME after the astrisk (*) has

been replaced with the query name. The purpose for this special form is query logging in the walled garden's authority DNS server.

All of the actions specified in all of the individual records in a policy zone can be overridden with a `policy` clause in the `response-policy` option. An organization using a policy zone provided by another organization might use this mechanism to redirect domains to its own walled garden.

GIVEN

The placeholder policy says "do not override but perform the action specified in the zone."

DISABLED

The testing override policy causes policy zone records to do nothing but log what they would have done if the policy zone were not disabled. The response to the DNS query will be written (or not) according to any triggered policy records that are not disabled. Disabled policy zones should appear first, because they will often not be logged if a higher precedence trigger is found first.

PASSTHRU; DROP; TCP-Only; NXDOMAIN; NODATA

override with the corresponding per-record policy.

CNAME domain

causes all RPZ policy records to act as if they were "cname domain" records.

By default, the actions encoded in a response policy zone are applied only to queries that ask for recursion (`RD=1`). That default can be changed for a single policy zone or all response policy zones in a view with a `recursive-only no` clause. This feature is useful for serving the same zone files both inside and outside an RFC 1918 cloud and using RPZ to delete answers that would otherwise contain RFC 1918 values on the externally visible name server or view.

Also by default, RPZ actions are applied only to DNS requests that either do not request DNSSEC metadata (`DO=0`) or when no DNSSEC records are available for request name in the original zone (not the response policy zone). This default can be changed for all response policy zones in a view with a `break-dnssec yes` clause. In that case, RPZ actions are applied regardless of DNSSEC. The name of the clause option reflects the fact that results rewritten by RPZ actions cannot verify.

No DNS records are needed for a QNAME or Client-IP trigger. The name or IP address itself is sufficient, so in principle the query name need not be recursively resolved. However, not resolving the requested name can leak the fact that response policy rewriting is in use and that the name is listed in a policy zone to operators of servers for listed names. To prevent that information leak, by default any recursion needed for a request is done before any policy triggers are considered. Because listed domains often have slow authoritative servers, this default behavior can cost significant time. The `qname-wait-recurse no` option overrides that default behavior when recursion cannot change a non-error response. The option does not affect QNAME or client-IP triggers in policy zones listed after other zones containing IP, NSIP and NS-DNAME triggers, because those may depend on the A, AAAA, and NS records that would be found during recursive resolution. It also does not affect DNSSEC requests (`DO=1`) unless `break-dnssec yes` is in use, because the response would depend on

whether or not RRSIG records were found during resolution. Using this option can cause error responses such as SERVFAIL to appear to be rewritten, since no recursion is being done to discover problems at the authoritative server.

The TTL of a record modified by RPZ policies is set from the TTL of the relevant record in policy zone. It is then limited to a maximum value. The `max-policy-ttl` clause changes that maximum from its default of 5.

For example, you might use this option statement

```
response-policy { zone "badlist"; };
```

and this zone statement

```
zone "badlist" {type master; file "master/badlist"; allow-  
query {none;}; };
```

with this zone file

```
$TTL 1H  
@                               SOA  LOCALHOST.  named-mgr.example.com.  
    (1 1h 15m 30d 2h)  
    NS   LOCALHOST.  
  
; QNAME policy records.  There are no periods (.) after the  
owner names.  
nxdomain.domain.com      CNAME    .                ; NXDOMAIN  
policy  
*.nxdomain.domain.com    CNAME    .                ; NXDOMAIN  
policy  
nodata.domain.com        CNAME    *.              ; NODATA  
policy  
*.nodata.domain.com      CNAME    *.              ; NODATA  
policy  
bad.domain.com           A        10.0.0.1          ; redirect to  
a walled garden  
                        AAAA      2001:2::1  
bzone.domain.com         CNAME    garden.example.com.  
  
; do not rewrite (PASSTHRU) OK.DOMAIN.COM  
ok.domain.com            CNAME    rpz-passthru.  
  
; redirect x.bzone.domain.com to x.bzone.domain.com.garden.  
example.com  
*.bzone.domain.com       CNAME    *.garden.example.com.  
  
; IP policy records that rewrite all responses containing A  
records in 127/8
```

(continues on next page)

(continued from previous page)

```

;      except 127.0.0.1
8.0.0.0.127.rpz-ip      CNAME      .
32.1.0.0.127.rpz-ip      CNAME      rpz-passthru.

; NSDNAME and NSIP policy records
ns.domain.com.rpz-nsdname  CNAME      .
48.zz.2.2001.rpz-nsip      CNAME      .

; blacklist and whitelist some DNS clients
112.zz.2001.rpz-client-ip  CNAME      rpz-drop.
8.0.0.0.127.rpz-client-ip  CNAME      rpz-drop.

; force some DNS clients and responses in the example.com_
↪zone to TCP
16.0.0.1.10.rpz-client-ip  CNAME      rpz-tcp-only.
example.com                  CNAME      rpz-tcp-only.
*.example.com                CNAME      rpz-tcp-only.

```

RPZ can affect server performance. Each configured response policy zone requires the server to perform one to four additional database lookups before a query can be answered. For example, a DNS server with four policy zones, each with all four kinds of response triggers, QNAME, IP, NSIP, and NSDNAME, requires a total of 17 times as many database lookups as a similar DNS server with no response policy zones. A Loop server with adequate memory and one response policy zone with QNAME and IP triggers might achieve a maximum queries-per-second rate about 20% lower. A server with four response policy zones with QNAME and IP triggers might have a maximum QPS rate about 50% lower.

Responses rewritten by RPZ are counted in the `RPZRewrites` statistics.

6.32 Response Rate Limiting (RRL)

Excessive almost identical UDP *responses* can be controlled by configuring a `rate-limit` clause in an `options` or `view` statement. This mechanism keeps authoritative Loop from being used in amplifying reflection denial of service (DoS) attacks. Short truncated (TC=1) responses can be sent to provide rate-limited responses to legitimate clients within a range of forged, attacked IP addresses. Legitimate clients react to dropped or truncated response by retrying with UDP or with TCP respectively.

This mechanism is intended for authoritative DNS servers. It can be used on recursive servers but can slow applications such as SMTP servers (mail receivers) and HTTP clients (web browsers) that repeatedly request the same domains. When possible, closing "open" recursive servers is better.

Response rate limiting uses a "credit" or "token bucket" scheme. Each combination of identical response and client has a conceptual account that earns a specified number of credits every second. A prospective response debits its account by one. Responses are dropped or truncated while the account is negative. Responses are tracked within

a rolling window of time which defaults to 15 seconds, but can be configured with the `window` option to any value from 1 to 3600 seconds (1 hour). The account cannot become more positive than the per-second limit or more negative than `window` times the per-second limit. When the specified number of credits for a class of responses is set to 0, those responses are not rate limited.

The notions of "identical response" and "DNS client" for rate limiting are not simplistic. All responses to an address block are counted as if to a single client. The prefix lengths of addresses blocks are specified with `ipv4-prefix-length` (default 24) and `ipv6-prefix-length` (default 56).

All non-empty responses for a valid domain name (`qname`) and record type (`qtype`) are identical and have a limit specified with `responses-per-second` (default 0 or no limit). All empty (NODATA) responses for a valid domain, regardless of query type, are identical. Responses in the NODATA class are limited by `nodata-per-second` (default `responses-per-second`). Requests for any and all undefined subdomains of a given valid domain result in NXDOMAIN errors, and are identical regardless of query type. They are limited by `nxdomains-per-second` (default `base responses-per-second`). This controls some attacks using random names, but can be relaxed or turned off (set to 0) on servers that expect many legitimate NXDOMAIN responses, such as from anti-spam blacklists. Referrals or delegations to the server of a given domain are identical and are limited by `referrals-per-second` (default `responses-per-second`).

Responses generated from local wildcards are counted and limited as if they were for the parent domain name. This controls flooding using `random.wild.example.com`.

All requests that result in DNS errors other than NXDOMAIN, such as SERVFAIL and FORMERR, are identical regardless of requested name (`qname`) or record type (`qtype`). This controls attacks using invalid requests or distant, broken authoritative servers. By default the limit on errors is the same as the `responses-per-second` value, but it can be set separately with `errors-per-second`.

Many attacks using DNS involve UDP requests with forged source addresses. Rate limiting prevents the use of Loop to flood a network with responses to requests with forged source addresses, but could let a third party block responses to legitimate requests. There is a mechanism that can answer some legitimate requests from a client whose address is being forged in a flood. Setting `slip` to 2 (its default) causes every other UDP request to be answered with a small truncated (TC=1) response. The small size and reduced frequency, and so lack of amplification, of "slipped" responses make them unattractive for reflection DoS attacks. `slip` must be between 0 and 10. A value of 0 does not "slip": no truncated responses are sent due to rate limiting, all responses are dropped. A value of 1 causes every response to slip; values between 2 and 10 cause every *n*'th response to slip. Some error responses including REFUSED and SERVFAIL cannot be replaced with truncated responses and are instead leaked at the `slip` rate.

(NOTE: Dropped responses from an authoritative server may reduce the difficulty of a third party successfully forging a response to a recursive resolver. The best security against forged responses is for authoritative operators to sign their zones using DNSSEC and for resolver operators to validate the responses. When this is not an option, operators who are more concerned with response integrity than with flood

mitigation may consider setting `slip` to 1, causing all rate-limited responses to be truncated rather than dropped. This reduces the effectiveness of rate-limiting against reflection attacks.)

When the approximate query per second rate exceeds the `qps-scale` value, then the `responses-per-second`, `errors-per-second`, `nxdomains-per-second` and `all-per-second` values are reduced by the ratio of the current rate to the `qps-scale` value. This feature can tighten defenses during attacks. For example, with `qps-scale 250`; `responses-per-second 20`; and a total query rate of 1000 queries/second for all queries from all DNS clients including via TCP, then the effective responses/second limit changes to $(250/1000)*20$ or 5. Responses sent via TCP are not limited but are counted to compute the query per second rate.

Communities of DNS clients can be given their own parameters or no rate limiting by putting `rate-limit` statements in view statements instead of the global `option` statement. A `rate-limit` statement in a view replaces, rather than supplementing, a `rate-limit` statement among the main options. DNS clients within a view can be exempted from rate limits with the `exempt-clients` clause.

UDP responses of all kinds can be limited with the `all-per-second` phrase. This rate limiting is unlike the rate limiting provided by `responses-per-second`, `errors-per-second`, and `nxdomains-per-second` on a DNS server which are often invisible to the victim of a DNS reflection attack. Unless the forged requests of the attack are the same as the legitimate requests of the victim, the victim's requests are not affected. Responses affected by an `all-per-second` limit are always dropped; the `slip` value has no effect. An `all-per-second` limit should be at least 4 times as large as the other limits, because single DNS clients often send bursts of legitimate requests. For example, the receipt of a single mail message can prompt requests from an SMTP server for NS, PTR, A, and AAAA records as the incoming SMTP/TCP/IP connection is considered. The SMTP server can need additional NS, A, AAAA, MX, TXT, and SPF records as it considers the `SMTP Mail From` command. Web browsers often repeatedly resolve the same names that are repeated in HTML `` tags in a page. `All-per-second` is similar to the rate limiting offered by firewalls but often inferior. Attacks that justify ignoring the contents of DNS responses are likely to be attacks on the DNS server itself. They usually should be discarded before the DNS server spends resources making TCP connections or parsing DNS requests, but that rate limiting must be done before the DNS server sees the requests.

The maximum size of the table used to track requests and rate limit responses is set with `max-table-size`. Each entry in the table is between 40 and 80 bytes. The table needs approximately as many entries as the number of requests received per second. The default is 20,000. To reduce the cold start of growing the table, `min-table-size` (default 500) can set the minimum table size. Enable `rate-limit` category logging to monitor expansions of the table and inform choices for the initial and maximum table size.

Use `log-only yes` to test rate limiting parameters without actually dropping any requests.

Responses dropped by rate limits are included in the `RateDropped` and `QryDropped` statistics. Responses that truncated by rate limits are included in `RateSlipped` and

RespTruncated.

6.32.1 Dynamic Update Policies

Loop supports two alternative methods of granting clients the right to perform dynamic updates to a zone, configured by the `allow-update` and `update-policy` option, respectively.

The `allow-update` clause is a simple access control list. Any client that matches the ACL is granted permission to update any record in the zone.

The `update-policy` clause allows more fine-grained control over what updates are allowed. It specifies a set of rules, in which each rule either grants or denies permission for one or more names in the zone to be updated by one or more identities. Identity is determined by the key that signed the update request using either TSIG or SIG(0). In most cases, `update-policy` rules only apply to key-based identities. There is no way to specify update permissions based on client source address.

`update-policy` rules are only meaningful for zones of type `master`, and are not allowed in any other zone type. It is a configuration error to specify both `allow-update` and `update-policy` at the same time.

A pre-defined `update-policy` rule can be switched on with the command `update-policy local;`. Using this in a zone causes `named` to generate a TSIG session key when starting up and store it in a file; this key can then be used by local clients to update the zone while `named` is running. By default, the session key is stored in the file `/var/run/loop/session.key`, the key name is "local-ddns", and the key algorithm is HMAC-SHA256. These values are configurable with the `session-keyfile`, `session-keyname` and `session-keyalg` options, respectively. A client running on the local system, if run with appropriate permissions, may read the session key from the key file and use it to sign update requests. The zone's update policy will be set to allow that key to change any record within the zone. Assuming the key name is "local-ddns", this policy is equivalent to:

```
update-policy { grant local-ddns zonesub any; };
```

...with the additional restriction that only clients connecting from the local system will be permitted to send updates.

Note that only one session key is generated by `named`; all zones configured to use `update-policy local` will accept the same key.

The command `nsupdate -l` implements this feature, sending requests to localhost and signing them using the key retrieved from the session key file.

Other rule definitions look like this:

```
( grant | deny ) identity ruletype name types
```

Each rule grants or denies privileges. Rules are checked in the order in which they are specified in the `update-policy` statement. Once a message has successfully matched a rule, the operation is immediately granted or denied, and no further rules

are examined. There are 10 types of rules; the rule type is specified by the `ruletype` field, and the interpretation of other fields varies depending on the rule type.

In general, a rule is matched when the key that signed an update request matches the `identity` field, the name of the record to be updated matches the `name` field (in the manner specified by the `ruletype` field), and the type of the record to be updated matches the `types` field. Details for each rule type are described below.

The `identity` field must be set to a fully-qualified domain name. In most cases, this represents the name of the TSIG or SIG(0) key that must be used to sign the update request. If the specified name is a wildcard, it is subject to DNS wildcard expansion, and the rule may apply to multiple identities. When a TKEY exchange has been used to create a shared secret, the identity of the key used to authenticate the TKEY exchange will be used as the identity of the shared secret. Some rule types use identities matching the client's Kerberos principal (e.g, "host/machine@REALM") or Windows realm (machine\$@REALM).

The `name` field also specifies a fully-qualified domain name. This often represents the name of the record to be updated. Interpretation of this field is dependent on rule type.

If no `types` are explicitly specified, then a rule matches all types except RRSIG, NS, SOA, NSEC and NSEC3. Types may be specified by name, including "ANY" (ANY matches all types except NSEC and NSEC3, which can never be updated). Note that when an attempt is made to delete all records associated with a name, the rules are checked for each existing record type.

The `ruletype` field has 10 values: `name`, `subdomain`, `wildcard`, `self`, `selfsub`, `selfwild`, `tcp-self`, `6to4-self`, `zonesub`, and `external`.

Name	Description
name	Exact-match semantics. This rule matches when the name being updated is identical to the contents of the name field.
subdomain	This rule matches when the name being updated is a subdomain of, or identical to, the contents of the name field.
zonesub	This rule is similar to subdomain, except that it matches when the name being updated is a subdomain of the zone in which the update-policy statement appears. This obviates the need to type the zone name twice, and enables the use of a standard update-policy statement in multiple zones without modification. When this rule is used, the name field is omitted.
wildcard	The name field is subject to DNS wildcard expansion, and this rule matches when the name being updated is a valid expansion of the wildcard.
self	This rule matches when the name of the record being updated matches the contents of the identity field. The name field is ignored. To avoid confusion, it is recommended that this field be set to the same value as the identity field or to . The self rule type is most useful when allowing one key per name to update, where the key has the same name as the record to be updated. In this case, the identity field can be specified as * (an asterisk).
selfsub	This rule is similar to self except that subdomains of self can also be updated.
selfwild	This rule is similar to self except that only subdomains of self can be updated.
tcp-self	This rule allows updates that have been sent via TCP and for which the standard mapping from the client's IP address into the in-addr.arpa and ip6.arpa namespaces match the name to be updated. The identity field must match that name. The name field should be set to .. Note that, since identity is based on the client's IP address, it is not necessary for update request messages

6.32.2 Multiple views

When multiple views are in use, a zone may be referenced by more than one of them. Often, the views will contain different zones with the same name, allowing different clients to receive different answers for the same queries. At times, however, it is desirable for multiple views to contain identical zones. The `in-view` zone option provides an efficient way to do this: it allows a view to reference a zone that was defined in a previously configured view. Example:

```
view internal {
    match-clients { 10/8; };

    zone example.com {
        type master;
        file "example-external.db";
    };
};

view external {
    match-clients { any; };

    zone example.com {
        in-view internal;
    };
};
```

An `in-view` option cannot refer to a view that is configured later in the configuration file.

A zone statement which uses the `in-view` option may not use any other options with the exception of `forward` and `forwarders`. (These options control the behavior of the containing view, rather than changing the zone object itself.)

Zone level acls (e.g. `allow-query`, `allow-transfer`) and other configuration details of the zone are all set in the view the referenced zone is defined in. Care need to be taken to ensure that acls are wide enough for all views referencing the zone.

An `in-view` zone cannot be used as a response policy zone.

An `in-view` zone is not intended to reference a `forward` zone.

MASTER FILES

7.1 Types of Resource Records and When to Use Them

This section, largely borrowed from RFC 1034, describes the concept of a Resource Record (RR) and explains when each is used. Since the publication of RFC 1034, several new RRs have been identified and implemented in the DNS. These are also included.

7.1.1 Resource Records

A domain name identifies a node. Each node has a set of resource information, which may be empty. The set of resource information associated with a particular name is composed of separate RRs. The order of RRs in a set is not significant and need not be preserved by name servers, resolvers, or other parts of the DNS. However, sorting of multiple RRs is permitted for optimization purposes, for example, to specify that a particular nearby server be tried first. See *section_title* and *RRset ordering*.

The components of a Resource Record are:

owne name	The domain name where the RR is found.
type	An encoded 16-bit value that specifies the type of the resource record.
TTL	The time-to-live of the RR. This field is a 32-bit integer in units of seconds, and is primarily used by resolvers when they cache RRs. The TTL describes how long a RR can be cached before it should be discarded.
class	An encoded 16-bit value that identifies a protocol family or instance of a protocol.
RDAI	The resource data. The format of the data is type (and sometimes class) specific.

The following are *types* of valid RRs:

A	A host address. In the IN class, this is a 32-bit IP address. Described in RFC 1035
AAAA	IPv6 address. Described in RFC 1886.
A6	IPv6 address. This can be a partial address (a suffix) and an indirection to the na

AFSDB	Location of AFS database servers. Experimental. Described in RFC 1183.
APL	Address prefix list. Experimental. Described in RFC 3123.
ATMA	ATM Address.
AVC	Application Visibility and Control record.
CAA	Identifies which Certificate Authorities can issue certificates for this domain and
CDNSKEY	Identifies which DNSKEY records should be published as DS records in the parent zone.
CDS	Contains the set of DS records that should be published by the parent zone.
CERT	Holds a digital certificate. Described in RFC 2538.
CNAME	Identifies the canonical name of an alias. Described in RFC 1035.
CSYNC	Child-to-Parent Synchronization in DNS as described in RFC 7477.
DHCID	Is used for identifying which DHCP client is associated with this name. Described in RFC 4034.
DLV	A DNS Look-aside Validation record which contains the records that are used as
DNAME	Replaces the domain name specified with another name to be looked up, effectively
DNSKEY	Stores a public key associated with a signed DNS zone. Described in RFC 4034.
DOA	Implements the Digital Object Architecture over DNS. Experimental.
DS	Stores the hash of a public key associated with a signed DNS zone. Described in RFC 4034.
EID	End Point Identifier.
EUI48	A 48-bit EUI address. Described in RFC 7043.
EUI64	A 64-bit EUI address. Described in RFC 7043.
GID	Reserved.
GPOS	Specifies the global position. Superseded by LOC.
HINFO	Identifies the CPU and OS used by a host. Described in RFC 1035.
HIP	Host Identity Protocol Address. Described in RFC 5205.
IPSECKEY	Provides a method for storing IPsec keying material in DNS. Described in RFC 4025.
ISDN	Representation of ISDN addresses. Experimental. Described in RFC 1183.
KEY	Stores a public key associated with a DNS name. Used in original DNSSEC; replaced by DNSKEY.
KX	Identifies a key exchanger for this DNS name. Described in RFC 2230.
L32	Holds 32-bit Locator values for Identifier-Locator Network Protocol. Described in RFC 6742.
L64	Holds 64-bit Locator values for Identifier-Locator Network Protocol. Described in RFC 6742.
LOC	For storing GPS info. Described in RFC 1876. Experimental.
LP	Identifier-Locator Network Protocol. Described in RFC 6742.
MB	Mail Box. Historical.
MD	Mail Destination. Historical.
MF	Mail Forwarder. Historical.
MG	Mail Group. Historical.
MINFO	Mail Information.
MR	Mail Rename. Historical.
MX	Identifies a mail exchange for the domain with a 16-bit preference value (lower is preferred).
NAPTR	Name authority pointer. Described in RFC 2915.
NID	Holds values for Node Identifiers in Identifier-Locator Network Protocol. Described in RFC 6742.
NINFO	Contains zone status information.
NIMLOC	Nimrod Locator.
NSAP	A network service access point. Described in RFC 1706.
NSAP-PTR	Historical.
NS	The authoritative name server for the domain. Described in RFC 1035.
NSEC	Used in DNSSECbis to securely indicate that RRs with an owner name in a certain zone do not exist.

NSEC3	Used in DNSSECbis to securely indicate that RRs with an owner name in a certain
NSEC3PARAM	Used in DNSSECbis to tell the authoritative server which NSEC3 chains are available
NULL	This is an opaque container.
NXT	Used in DNSSEC to securely indicate that RRs with an owner name in a certain
OPENPGPKEY	Used to hold an OPENPGPKEY.
PTR	A pointer to another part of the domain name space. Described in RFC 1035.
PX	Provides mappings between RFC 822 and X.400 addresses. Described in RFC 2163.
RKEY	Resource key.
RP	Information on persons responsible for the domain. Experimental. Described in RFC 4034.
RRSIG	Contains DNSSECbis signature data. Described in RFC 4034.
RT	Route-through binding for hosts that do not have their own direct wide area network
SIG	Contains DNSSEC signature data. Used in original DNSSEC; replaced by RRSIG.
SINK	The kitchen sink record.
SMIMEA	The S/MIME Security Certificate Association.
SOA	Identifies the start of a zone of authority. Described in RFC 1035.
SPF	Contains the Sender Policy Framework information for a given email domain. Described in RFC 7208.
SRV	Information about well known network services (replaces WKS). Described in RFC 2782.
SSHFP	Provides a way to securely publish a secure shell key's fingerprint. Described in RFC 4255.
TA	Trust Anchor. Experimental.
TALINK	Trust Anchor Link. Experimental.
TLSA	Transport Layer Security Certificate Association. Described in RFC 6698.
TXT	Text records. Described in RFC 1035.
UID	Reserved.
UINFO	Reserved.
UNSPEC	Reserved. Historical.
URI	Holds a URI. Described in RFC 7553.
WKS	Information about which well known network services, such as SMTP, that a domain
X25	Representation of X.25 network addresses. Experimental. Described in RFC 1183.

The following *classes* of resource records are currently valid in the DNS:

IN	The Internet.
CH	Chaosnet, a LAN protocol created at MIT in the mid-1970s. Rarely used for its historical purpose, but reused for Loop's built-in server information zones, e.g., <code>version.loop</code> .
HS	Hesiod, an information service developed by MIT's Project Athena. It is used to share information about various systems databases, such as users, groups, printers and so on.

The owner name is often implicit, rather than forming an integral part of the RR. For example, many name servers internally form tree or hash structures for the name space, and chain RRs off nodes. The remaining RR parts are the fixed header (type, class, TTL) which is consistent for all RRs, and a variable part (RDATA) that fits the needs of the resource being described.

The meaning of the TTL field is a time limit on how long an RR can be kept in a cache.

This limit does not apply to authoritative data in zones; it is also timed out, but by the refreshing policies for the zone. The TTL is assigned by the administrator for the zone where the data originates. While short TTLs can be used to minimize caching, and a zero TTL prohibits caching, the realities of Internet performance suggest that these times should be on the order of days for the typical host. If a change can be anticipated, the TTL can be reduced prior to the change to minimize inconsistency during the change, and then increased back to its former value following the change.

The data in the RDATA section of RRs is carried as a combination of binary strings and domain names. The domain names are frequently used as "pointers" to other data in the DNS.

7.1.2 Textual expression of RRs

RRs are represented in binary form in the packets of the DNS protocol, and are usually represented in highly encoded form when stored in a name server or resolver. In the examples provided in RFC 1034, a style similar to that used in master files was employed in order to show the contents of RRs. In this format, most RRs are shown on a single line, although continuation lines are possible using parentheses.

The start of the line gives the owner of the RR. If a line begins with a blank, then the owner is assumed to be the same as that of the previous RR. Blank lines are often included for readability.

Following the owner, we list the TTL, type, and class of the RR. Class and type use the mnemonics defined above, and TTL is an integer before the type field. In order to avoid ambiguity in parsing, type and class mnemonics are disjoint, TTLs are integers, and the type mnemonic is always last. The IN class and TTL values are often omitted from examples in the interests of clarity.

The resource data or RDATA section of the RR are given using knowledge of the typical representation for the data.

For example, we might show the RRs carried in a message as:

ISI.EDU.	MX	10	VENERA.ISI.EDU.
	MX	10	VAXA.ISI.EDU
VENERA.ISI.EDU	A	128.9.0.32	
	A	10.1.0.52	
VAXA.ISI.EDU	A	10.2.0.27	
	A	128.9.0.33	

The MX RRs have an RDATA section which consists of a 16-bit number followed by a domain name. The address RRs use a standard IP address format to contain a 32-bit internet address.

The above example shows six RRs, with two RRs at each of three domain names.

Similarly we might see:

XX.LCS.MIT.EDU.	IN	A	10.0.0.44
	CH	A	MIT.EDU. 2420

This example shows two addresses for `XX.LCS.MIT.EDU`, each of a different class.

7.2 Discussion of MX Records

As described above, domain servers store information as a series of resource records, each of which contains a particular piece of information about a given domain name (which is usually, but not always, a host). The simplest way to think of a RR is as a typed pair of data, a domain name matched with a relevant datum, and stored with some additional type information to help systems determine when the RR is relevant.

MX records are used to control delivery of email. The data specified in the record is a priority and a domain name. The priority controls the order in which email delivery is attempted, with the lowest number first. If two priorities are the same, a server is chosen randomly. If no servers at a given priority are responding, the mail transport agent will fall back to the next largest priority. Priority numbers do not have any absolute meaning — they are relevant only relative to other MX records for that domain name. The domain name given is the machine to which the mail will be delivered. It *must* have an associated address record (A or AAAA) — CNAME is not sufficient.

For a given domain, if there is both a CNAME record and an MX record, the MX record is in error, and will be ignored. Instead, the mail will be delivered to the server specified in the MX record pointed to by the CNAME. For example:

example.com.	IN	MX	10	mail.example.com.
	IN	MX	10	mail2.example.com.
	IN	MX	20	mail.backup.org.
mail.example.com.	IN	A	10.0.0.1	
mail2.example.com.	IN	A	10.0.0.2	

Mail delivery will be attempted to `mail.example.com` and `mail2.example.com` (in any order), and if neither of those succeed, delivery to `mail.backup.org` will be attempted.

7.3 Setting TTLs

The time-to-live (TTL) field of a resource record (RR) is a 32-bit integer represented in units of seconds, and is primarily used by resolvers when they cache RRs. The TTL describes how long a RR can be cached before it should be discarded. The following three types of TTL are currently used in a zone file.

SOA

The last field in the SOA RDATA is the negative cache TTL. This controls how

long caching resolvers will cache negative responses. The maximum time for negative caching is 3 hours (3h).

\$TTL

The \$TTL directive at the top of the zone file (before the SOA) gives a default TTL for every RR without a specific TTL set.

RR TTLs

Each RR can have a TTL as the second field in the RR, which will control how long other servers can cache it.

All of these TTLs default to units of seconds, though units can be explicitly specified, for example, 1h30m.

7.4 Inverse Mapping in IPv4

Reverse name resolution (that is, translation from IP address to name) is achieved by means of the *in-addr.arpa* domain and PTR records. Entries in the in-addr.arpa domain are made in least-to-most significant order, read left to right. This is the opposite order to the way IP addresses are usually written. Thus, a machine with an IP address of 10.1.2.3 would have a corresponding in-addr.arpa name of 3.2.1.10.in-addr.arpa. This name should have a PTR resource record whose data field is the name of the machine or, optionally, multiple PTR records if the machine has more than one name. For example, in the [example.com] domain:

\$ORIGIN	2.1.10.in-addr.arpa
3	IN PTR foo.example.com.

Note

The \$ORIGIN lines in the examples are for providing context to the examples only--they do not necessarily appear in the actual usage. They are only used here to indicate that the example is relative to the listed origin.

7.5 Directives

The zone master file format was initially defined in [RFC 1035](#) and has subsequently been extended. While the master file format itself is class independent, all records in a master file must be of the same RR class.

7.5.1 @

When used in the label (or name) field, the asperand or at-sign (@) symbol represents the current origin. At the start of the zone file, it is the <zone_name> (followed by trailing dot).

7.5.2 \$ORIGIN

Syntax: `$ORIGIN domain-name [comment]`

`$ORIGIN` sets the domain name that will be appended to any unqualified records. When a zone is first read in there is an implicit `$ORIGIN <zone_name>.` (followed by trailing dot). The current `$ORIGIN` is appended to the domain specified in the `$ORIGIN` argument if it is not absolute.

```
$ORIGIN example.com.
WWW      CNAME      MAIN-SERVER
```

is equivalent to

```
WWW.EXAMPLE.COM. CNAME MAIN-SERVER.EXAMPLE.COM.
```

7.5.3 \$INCLUDE

Syntax: `$INCLUDE filename [origin] [comment]`

Read and process the file `filename` as if it were included into the file at this point. If `origin` is specified the file is processed with `$ORIGIN` set to that value, otherwise the current `$ORIGIN` is used.

The origin and the current domain name revert to the values they had prior to the `$INCLUDE` once the file has been read.

Note

RFC 1035 specifies that the current origin should be restored after an `$INCLUDE`, but it is silent on whether the current domain name should also be restored. Loop restores both of them. This could be construed as a deviation from RFC 1035, a feature, or both.

7.5.4 \$TTL

Syntax: `$TTL default-ttl [comment]`

Set the default Time To Live (TTL) for subsequent records with undefined TTLs. Valid TTLs are of the range 0-2147483647 seconds.

`$TTL` is defined in RFC 2308.

7.5.5 \$GENERATE

Syntax: `$GENERATE range lhs [ttl] [class] type rhs [comment]`

`$GENERATE` is used to create a series of resource records that only differ from each other by an iterator. `$GENERATE` can be used to easily generate the sets of records required to support sub /24 reverse delegations described in RFC 2317: Classless IN-ADDR.ARPA delegation.

```
$ORIGIN 0.0.192.IN-ADDR.ARPA.  
$GENERATE 1-2 @ NS SERVER$.EXAMPLE.  
$GENERATE 1-127 $ CNAME $.0
```

is equivalent to

```
0.0.0.192.IN-ADDR.ARPA. NS SERVER1.EXAMPLE.  
0.0.0.192.IN-ADDR.ARPA. NS SERVER2.EXAMPLE.  
1.0.0.192.IN-ADDR.ARPA. CNAME 1.0.0.0.192.IN-ADDR.ARPA.  
2.0.0.192.IN-ADDR.ARPA. CNAME 2.0.0.0.192.IN-ADDR.ARPA.  
...  
127.0.0.192.IN-ADDR.ARPA. CNAME 127.0.0.0.192.IN-ADDR.ARPA.
```

Generate a set of A and MX records. Note the MX's right hand side is a quoted string. The quotes will be stripped when the right hand side is processed.

```
$ORIGIN EXAMPLE.  
$GENERATE 1-127 HOST-$ A 1.2.3.$  
$GENERATE 1-127 HOST-$ MX "0 ."
```

is equivalent to

```
HOST-1.EXAMPLE. A 1.2.3.1  
HOST-1.EXAMPLE. MX 0 .  
HOST-2.EXAMPLE. A 1.2.3.2  
HOST-2.EXAMPLE. MX 0 .  
HOST-3.EXAMPLE. A 1.2.3.3  
HOST-3.EXAMPLE. MX 0 .  
...  
HOST-127.EXAMPLE. A 1.2.3.127  
HOST-127.EXAMPLE. MX 0 .
```

<code>range</code>	This can be one of two forms: start-stop or start-stop/step. If the first form is used, then step is set to 1. start, stop and step must be positive integers between 0 and $(2^{31})-1$. start must not be larger than stop.
<code>lhs</code>	<p>This describes the owner name of the resource records to be created. Any single \$ (dollar sign) symbols within the <code>lhs</code> string are replaced by the iterator value. To get a \$ in the output, you need to escape the \$ using a backslash \, e.g. \\$. The \$ may optionally be followed by modifiers which change the offset from the iterator, field width and base. Modifiers are introduced by a { (left brace) immediately following the \$ as <code>\${offset[,width[,base]]}</code>. For example, <code>\${-20,3,d}</code> subtracts 20 from the current value, prints the result as a decimal in a zero-padded field of width 3. Available output forms are decimal (d), octal (o), hexadecimal (x or X for uppercase) and nibble (n or N\ for uppercase). The default modifier is <code>\${0,0,d}</code>. If the <code>lhs</code> is not absolute, the current <code>\$ORIGIN</code> is appended to the name.</p> <p>In nibble mode the value will be treated as if it was a reversed hexadecimal string with each hexadecimal digit as a separate label. The width field includes the label separator.</p> <p>For compatibility with earlier versions, <code>\$\$</code> is still recognized as indicating a literal \$ in the output.</p>
<code>ttd</code>	<p>Specifies the time-to-live of the generated records. If not specified this will be inherited using the normal TTL inheritance rules.</p> <p><code>class</code> and <code>ttd</code> can be entered in either order.</p>
<code>class</code>	<p>Specifies the class of the generated records. This must match the zone class if it is specified.</p> <p><code>class</code> and <code>ttd</code> can be entered in either order.</p>
<code>type</code>	Any valid type.
<code>rhs</code>	<code>rhs</code> , optionally, quoted string.

Note

The `$GENERATE` directive is a Loop extension and not part of the standard zone master file format.

STATISTICS

The Loop nameserver maintains many statistics counters and provides an interface for administrators to access them in an XML format.

Warning

The counter symbol names may change in future major releases. Please check the release notes before upgrading.

The statistics information is categorized into the following sections.

In-coming Requests	The number of incoming DNS requests for each OPCODE.
In-coming Queries	The number of incoming queries for each RR type.
Out-going Queries	The number of outgoing queries for each RR type sent from the internal resolver. Maintained per view.
Name Server Statistics	Statistics counters about incoming request processing.
Zone Maintenance Statistics	Statistics counters regarding zone maintenance operations such as zone transfers.
Resolver Statistics	Statistics counters about name resolution performed in the internal resolver. Maintained per view.
Cache DB RRsets	The number of RRsets per RR type and nonexistent names stored in the cache database. If the exclamation mark (!) is printed for a RR type, it means that particular type of RRset is known to be nonexistent (this is also known as "NXRRSET"). If a hash mark (#) is present then the RRset is marked for garbage collection. Maintained per view.
Socket I/O Statistics	Statistics counters about network related events.

A subset of statistics is collected and shown per-zone for which the server has the authority when `zone-statistics` is set to `full` (or `yes` for backward compatibility. See the description of `zone-statistics` in *section_title* for further details.

These statistics counters are shown with their zone and view names. The view name is omitted when the server is not configured with explicit views.

Statistics counters are dumped in an XML format to the file specified by the `statistics-file` configuration option.

8.1 XML Format

Error

TODO: Add examples of parsing the XML statistics.

8.2 Counters

The following tables summarize statistics counters that Loop provides. For each row of the tables, the leftmost column is the abbreviated symbol name of that counter. These symbols are shown in the statistics information accessed via an HTTP statistics channel. The rightmost column gives the description of the counter, which is also shown in the statistics file (but, in this document, possibly with slight modification for better readability). Additional notes may also be provided in this column.

8.3 Nameserver Counters

Symbol	Description
Requestv4	IPv4 requests received. Note: this also counts non query requests.
Requestv6	IPv6 requests received. Note: this also counts non query requests.
ReqEdns0	Requests with EDNS(0) received.
ReqBadEDNSVer	Requests with unsupported EDNS version received.
ReqTSIG	Requests with TSIG received.
ReqSIG0	Requests with SIG(0) received.
ReqBadSIG	Requests with invalid (TSIG or SIG(0)) signature.
ReqTCP	TCP requests received.
AuthQryRej	Authoritative (non recursive) queries rejected.
RecQryRej	Recursive queries rejected.
XfrRej	Zone transfer requests rejected.
UpdateRej	Dynamic update requests rejected.
Response	Responses sent.
RespTruncated	Truncated responses sent.
RespEDNS0	Responses with EDNS(0) sent.
RespTSIG	Responses with TSIG sent.
RespSIG0	Responses with SIG(0) sent.

continues on next page

Table 1 – continued from previous page

Symbol	Description
QrySuccess	Queries resulted in a successful answer. This means the query which returns a NOERROR response with at least one answer RR.
QryAuthAns	Queries resulted in authoritative answer.
QryNoauthAns	Queries resulted in non authoritative answer.
QryReferral	Queries resulted in referral answer.
QryNxrrset	Queries resulted in NOERROR responses with no data.
QrySERVFAIL	Queries resulted in SERVFAIL.
QryFORMERR	Queries resulted in FORMERR.
QryNXDOMAIN	Queries resulted in NXDOMAIN.
QryRecursion	Queries which caused the server to perform recursion in order to find the final answer.
QryDuplicate	Queries which the server attempted to recurse but discovered an existing query with the same IP address, port, query ID, name, type and class already being processed.
QryDropped	Recursive queries for which the server discovered an excessive number of existing recursive queries for the same name, type and class and were subsequently dropped. This is the number of dropped queries due to the reason explained with the <code>clients-per-query</code> and <code>max-clients-per-query</code> options (see the description about <code><#clients-per-query>`__.</code>)
QryFailure	Other query failures. This corresponds Note: this counter is provided mainly for backward compatibility with the previous versions. Normally a more fine-grained counters such as <code>AuthQryRej</code> and <code>RecQryRej</code> that would also fall into this counter are provided, and so this counter would not be of much interest in practice.
XfrReqDone	Requested zone transfers completed.
UpdateReqFwd	Update requests forwarded.
UpdateRespFwd	Update responses forwarded.
UpdateFwdFail	Dynamic update forward failed.

continues on next page

Table 1 – continued from previous page

Symbol	Description
UpdateDone	Dynamic updates completed.
UpdateFail	Dynamic updates failed.
UpdateBadPrereq	Dynamic updates rejected due to prerequisite failure.
RateDropped	Responses dropped by rate limits.
RateSlipped	Responses truncated by rate limits.
RPZRewrites	Response policy zone rewrites.

8.4 Zone Maintenance Counters

The following are statistics counters related to zone maintenance operations such as zone transfers.

<i>Symbol</i>	<i>Description</i>
NotifyOutv4	IPv4 notifies sent.
NotifyOutv6	IPv6 notifies sent.
NotifyInv4	IPv4 notifies received.
NotifyInv6	IPv6 notifies received.
NotifyRej	Incoming notifies rejected.
SOAOutv4	IPv4 SOA queries sent.
SOAOutv6	IPv6 SOA queries sent.
AXFRReqv4	IPv4 AXFR requested.
AXFRReqv6	IPv6 AXFR requested.
IXFRReqv4	IPv4 IXFR requested.
IXFRReqv6	IPv6 IXFR requested.
XfrSuccess	Zone transfer requests succeeded.
XfrFail	Zone transfer requests failed.

8.5 Resolver Counters

The following are statistics counters related to query resolution performed in the internal resolver. They are maintained per-view.

<i>Sym-</i>	<i>Description</i>
<i>bol</i>	
Que:	IPv4 queries sent.
Que:	IPv6 queries sent.
Res:	IPv4 responses received.
Res:	IPv6 responses received.
NXD:	NXDOMAIN received.
SER:	SERVFAIL received.
FOR:	FORMERR received.
Oth:	Other errors received.
EDN:	EDNS(0) query failures.
Mis:	Mismatch responses received. The DNS ID, response's source address, and/or the response's source port does not match what was expected. (The port must be 53 or as defined by the <code>port</code> option.) This may be an indication of a cache poisoning attempt.
Tru:	Truncated responses received.
Lam:	Lame delegations received.
Ret:	Query retries performed.
Que:	Queries aborted due to quota control.
Que:	Failures in opening query sockets. One common reason for such failures is a failure of opening a new socket due to a limitation on file descriptors.
Que:	Query timeouts.
Glue:	IPv4 NS address fetches invoked.
Glue:	IPv6 NS address fetches invoked.
Glue:	IPv4 NS address fetch failed.
Glue:	IPv6 NS address fetch failed.
Vali:	DNSSEC validation attempted.
Val:	DNSSEC validation succeeded.
Vali:	DNSSEC validation on negative information succeeded.
Vali:	DNSSEC validation failed.
Qry:	Frequency table on round trip times (RTTs) of queries. Each <code>nn</code> specifies the corresponding frequency. In the sequence of <code>nn_1</code> , <code>nn_2</code> , ..., <code>nn_m</code> , the value of <code>nn_i</code> is the number of queries whose RTTs are between <code>nn_(i-1)</code> (inclusive) and <code>nn_i</code> (exclusive) milliseconds. For the sake of convenience we define <code>nn_0</code> to be 0. The last entry should be represented as <code>nn_m+</code> , which means the number of queries whose RTTs are equal to or over <code>nn_m</code> milliseconds.

8.6 Socket I/O Counters

The following are statistics counters related to network events.

Symbol	Description
UDP4Open	IPv4 UDP sockets opened successfully
UDP6Open	IPv6 UDP sockets opened successfully

Table 2 – continued from previous page

Symbol	Description
TCP4Open	IPv4 TCP sockets opened successfully
TCP6Open	IPv6 TCP sockets opened successfully
RawOpen	Raw sockets opened successfully
UDP4OpenFail	Failures in opening IPv4 UDP sockets
UDP6OpenFail	Failures in opening IPv6 UDP sockets
TCP4OpenFail	Failures in opening IPv4 TCP sockets
TCP6OpenFail	Failures in opening IPv6 TCP sockets
RawOpenFail	Failures in opening raw sockets
UDP4Close	IPv4 UDP sockets closed
UDP6Close	IPv6 UDP sockets closed
TCP4Close	IPv4 TCP sockets closed
TCP6Close	IPv6 TCP sockets closed
RawClose	Raw sockets closed
UDP4BindFail	Failures in binding IPv4 UDP sockets
UDP6BindFail	Failures in binding IPv6 UDP sockets
TCP4BindFail	Failures in binding IPv4 TCP sockets
TCP6BindFail	Failures in binding IPv6 TCP sockets
UDP4ConnFail	Failures in connecting IPv4 UDP sockets
UDP6ConnFail	Failures in connecting IPv6 UDP sockets
TCP4ConnFail	Failures in connecting IPv4 TCP sockets
TCP6ConnFail	Failures in connecting IPv6 TCP sockets
UDP4Conn	IPv4 UDP connections established successfully
UDP6Conn	IPv6 UDP connections established successfully
TCP4Conn	IPv4 TCP connections established successfully
TCP6Conn	IPv6 TCP connections established successfully
TCP4AcceptFail	Failures in accepting incoming IPv4 TCP connection requests
TCP6AcceptFail	Failures in accepting incoming IPv6 TCP connection requests
TCP4Accept	Incoming IPv4 TCP connections successfully accepted
TCP6Accept	Incoming IPv6 TCP connections successfully accepted
UDP4SendErr	Errors in IPv4 UDP socket send operations
UDP6SendErr	Errors in IPv6 UDP socket send operations
TCP4SendErr	Errors in IPv4 TCP socket send operations
TCP6SendErr	Errors in IPv6 TCP socket send operations
UDP4RecvErr	Errors in IPv4 UDP socket receive operations. It includes errors of send operations.
UDP6RecvErr	Errors in IPv6 UDP socket receive operations. It includes errors of send operations.
TCP4RecvErr	Errors in IPv4 TCP socket receive operations
TCP6RecvErr	Errors in IPv6 TCP socket receive operations
RawRecvErr	Errors in raw socket receive operations
UDP4Active	IPv4 UDP sockets currently active
UDP6Active	IPv6 UDP sockets currently active
TCP4Active	IPv4 TCP sockets currently active
TCP6Active	IPv6 TCP sockets currently active
RawActive	Raw sockets currently active

SECURITY CONSIDERATIONS

9.1 Access Control Lists

Access Control Lists (ACLs) are address match lists that you can set up and nickname for future use in `allow-notify`, `allow-query`, `allow-query-on`, `allow-recursion`, `allow-recursion-on`, `block`, `allow-transfer`, etc.

Using ACLs allows you to have finer control over who can access your name server, without cluttering up your config files with huge lists of IP addresses.

It is a *good idea* to use ACLs, and to control access to your server. Limiting access to your server by outside parties can help prevent spoofing and denial of service (DoS) attacks against your server.

Here is an example of how to properly apply ACLs:

```
// Set up an ACL named "bogusnets" that will block
// RFC1918 space and some reserved space, which is
// commonly used in spoofing attacks.
acl bogusnets {
    0.0.0.0/8; 192.0.2.0/24; 224.0.0.0/3;
    10.0.0.0/8; 172.16.0.0/12; 192.168.0.0/16;
};

// Set up an ACL called our-nets. Replace this with the
// real IP numbers.
acl our-nets { x.x.x.x/24; x.x.x.x/21; };
options {
    ...
    ...
    allow-query { our-nets; };
    allow-recursion { our-nets; };
    ...
    block { bogusnets; };
    ...
};
```

(continues on next page)

(continued from previous page)

```
zone "example.com" {  
    type master;  
    file "m/example.com";  
    allow-query { any; };  
};
```

This allows recursive queries of the server from the outside unless recursion has been previously disabled.

9.2 `chroot()` and `setuid()`

It is possible to run **named** in a *chroot* environment (where it internally uses the *chroot(2)* function) by specifying the `-t` argument. This can help improve system security by placing **named** in a *sandbox*, which will limit the damage done if a server is compromised.

Another useful feature in Loop is the ability to run **named** as an unprivileged user (where it internally uses the *setuid(2)* function) by specifying the `-u` argument. We suggest running **named** as an unprivileged user when running in a *chroot* environment.

Here is an example command line to start **named** in a *chroot* sandbox `/var/lib/loop`, and `setuid()` it to user 202:

```
# /usr/sbin/named -u 202 -t /var/lib/loop
```

9.2.1 The *chroot* Environment

In order for a *chroot* environment to work properly in a particular directory (for example, `/var/lib/loop`), you will need to set up an environment that includes everything Loop needs to run. From Loop's point of view, `/var/lib/loop` is the root of the filesystem. You will need to adjust the values of options like `directory` and `pid-file` to account for this.

Depending on your operating system, you may need to set up things like `/dev/zero`, `/dev/random`, `/dev/log`, and `/etc/localtime`.

9.2.2 Using the *setuid* Function

Prior to running the **named** daemon, use the `touch` utility (to change file access and modification times) or the `chown` utility (to set the user id and/or group id) on files to which you want Loop to write.

Note

If the **named** daemon is running as an unprivileged user, it will not be able to bind to new restricted ports if the server is reloaded.

9.3 DNS UPDATE security

Access to the DNS UPDATE feature (dynamic updates) should be strictly controlled. Limiting access based on the IP address of the host requesting the update (by listing an IP address or network prefix in the `allow-update` zone option) is insecure as the source IP address of UDP datagrams can be forged. Also, if the IP addresses allowed by the `allow-update` option include the address of a slave server which performs forwarding of DNS UPDATES, the master can be trivially attacked by sending the DNS UPDATE to the slave, which will forward it to the master with its own source IP address causing the master to approve it without question.

For these reasons, we strongly recommend that DNS UPDATES be cryptographically authenticated only by means of transaction signatures (TSIG). That is, the `allow-update` option should list only TSIG key names, not IP addresses or network prefixes. Alternatively, the `update-policy` option can be used.

Some sites choose to keep all dynamically-updated DNS data in a subdomain, and delegate that subdomain to a separate child zone. This way, the parent zone which may contain critical data such as the IP addresses of public web and mail servers need not allow dynamic updates at all.

9.4 Control channel security

Control channel communications between **rndc** and **named** are cryptographically authenticated using HMACs to protect against unauthorized modification, but they are transmitted in-the-clear. Encryption is **not** currently used to hide control channel communications from interception. Depending on the requirement, an encrypted transport such as IPsec or Wireguard may be used when transmitting control channel communications over the public internet.

TROUBLESHOOTING

10.1 Common Problems

10.1.1 It's not working; how can I figure out what's wrong?

The best solution to solving installation and configuration issues is to take preventative measures by setting up logging files beforehand. The log files provide a source of hints and information that can be used to figure out what went wrong and how to fix the problem.

10.2 Incrementing and Changing the Serial Number

Zone serial numbers are just numbers — they aren't date related. A lot of people set them to a number that represents a date, usually of the form YYYYMMDDRR. Occasionally they will make a mistake and set them to a "date in the future" then try to correct them by setting them to the "current date". This causes problems because serial numbers are used to indicate that a zone has been updated. If the serial number on the slave server is lower than the serial number on the master, the slave server will attempt to update its copy of the zone.

Setting the serial number to a lower number on the master server than the slave server means that the slave will not perform updates to its copy of the zone.

The solution to this is to add 2147483647 ($2^{31}-1$) to the number, reload the zone and make sure all slaves have updated to the new zone serial number, then reset the number to what you want it to be, and reload the zone again.

RELEASE NOTES

This documentation corresponds to Loop version 1.99.7.20250831123527.772ce56e35.

11.1 Loop 1.99.7

The following are release notes for Loop 1.99.7:

- RT1428: The packaging of Loop was updated. Loop is part of a source code tree with other components such as Lease and Border, and previously only Loop and Lease were packaged. Other components of the tree are also packaged now as part of a multi-package RPM spec file. The `border-release` package which installed the `border-epel`, `border-epel-testing`, `border-fedora`, and `border-fedora-testing` DNF package repositories and their package signing PGP key has been replaced with the `akira-release` package which installs the `akira-epel`, `akira-epel-testing`, `akira-fedora`, and `akira-fedora-testing` DNF package repositories and their package signing PGP key.
- RT1660: Functionality to make **named** a daemon was moved to a new module in the libloop library. This is not a user-visible change.
- RT1661: `chroot(2)` functionality was moved to the libloop library. This is not a user-visible change.
- RT1662: Privilege dropping functionality (using Linux capabilities API) was moved to a new module in the libloop library. This is not a user-visible change.
- RT1657: `syslog` functionality was moved to a new module in the libloop library. This is not a user-visible change.
- RT1656: `ns_os_shutdownmsg()` was moved to live within the server module. This is not a user-visible change.
- RT1654: A `gethostname()` wrapper was moved to the libloop library. This is not a user-visible change.
- RT1653: PID file functionality was moved to a new module in the libloop library. This is not a user-visible change.
- RT1652: PID file code was refactored. This is not a user-visible change.

- RT1651: Error functions were refactored; `UNEXPECTED_ERROR` and `FATAL_ERROR` macros were removed. These are not user-visible changes.
- RT1648: `loop_lib_init()` now calls `tzset(3)`; it was previously called by the **named** program's code. This is not a user-visible change. The call may be removed in the future.
- RT1647: CPU count functionality was moved to a new module in the libloop library. This is not a user-visible change.
- RT1646: `NULL` is now allowed in `loop_mem_strdup()` which returns `NULL`. This is not a user-visible change.
- RT1640: A `loop_uuid` module was added. This is not a user-visible change.
- RT1639: A `loop_mime` module was added. This is not a user-visible change.
- RT1638: Support to create urlsafe Base64 was added to libloop (see [RFC 4648](#) section 5). This is not a user-visible change.
- RT1632: Some remnant code and documentation of the obsolete **named-compilezone** program were removed.
- RT1633: The Loop version string was incorrectly reported in some places previously due to some recent changes. Uses of `PACKAGE_` macros were changed so that the issue does not occur again.
- RT1613: All the programs of Loop and Lease were updated to use consistent command line options for version, verbosity, help, etc.
- RT1618: DNS names were added to the list of items in the data and privacy section of the Loop and Lease user manuals.

To upgrade from a previous release of Loop on an RPM platform, first remove the obsolete `border-release` and `loop-release` RPMs:

```
$ sudo dnf remove border-release loop-release
```

Then, install the `akira-release` package as described in the installation instructions (see the chapter titled [Installation](#)). This will install the new DNF package repositories.

Then, upgrade the `loop` package, which should install it from the new DNF package repository:

```
$ sudo dnf upgrade loop
```

You may clean up the old Border and Loop Package Signer keys using `clean-rpm-gpg-pubkey`:

```
$ sudo dnf install clean-rpm-gpg-pubkey
$ sudo clean-rpm-gpg-pubkey
```


11.2 Loop 1.99.6

The following are release notes for Loop 1.99.6:

- RT1606: The packaging of Loop was updated. Loop is part of a source code tree with other components such as Lease, and previously only Loop was packaged. Other components of the tree are also packaged now as part of a multi-package RPM spec file. The `loop-release` package which installed the `loop-epel`, `loop-epel-testing`, `loop-fedora`, and `loop-fedora-testing` DNF package repositories and their package signing PGP key has been replaced with the `border-release` package which installs the `border-epel`, `border-epel-testing`, `border-fedora`, and `border-fedora-testing` DNF package repositories and their package signing PGP key.
- RT1607: The system path for storing PID files, session key files, etc. was updated to use `runstatedir` instead of `localstatedir/run`, as RPM packaging makes a distinction in these paths and uses the former.
- RT1608: A documentation link was added to the Systemd `named.service` file.

To upgrade from a previous release of Loop on an RPM platform, first remove the obsolete `loop-release` RPM:

```
$ sudo dnf remove loop-release
```

Then, install the `border-release` package as described in the installation instructions (see the chapter titled [Installation](#)). This will install the new DNF package repositories.

Then, upgrade the `loop` package, which should install it from the new DNF package repository:

```
$ sudo dnf upgrade loop
```

You may clean up the old Loop Package Signer key using `clean-rpm-gpg-pubkey`:

```
$ sudo dnf install clean-rpm-gpg-pubkey
$ sudo clean-rpm-gpg-pubkey
```

11.3 Loop 1.99.5

The following are release notes for Loop 1.99.5:

- RT1577: Loop now applies a maximum RSA public exponent size of 256 bits during DNSSEC validation. Previously it was unlimited, and there is also no limit suggested in the DNS standards currently for the RSA public exponent. An unlimited RSA public exponent size can increase RSA signature validation times significantly. There was no vulnerability due to this lack of limit in practice, as other limits in buffer sizes and the OpenSSL library restricted the maximum exponent size that could be successfully used. However, these effects

were serendipitous, and an explicit limit of 256 bits has now been set as suggested in [FIPS 186-5](#) and [NIST SP 800-56B](#). The maximum allowed RSA public exponent size can be configured using the `max-rsa-exponent-size` option of `named.conf(5)`. 256 bits is also the maximum value allowed for the `max-rsa-exponent-size` config option. RSA DNSKEYs created by Loop continue to use a fixed public exponent 65537 which is 17 bits long.

- RT1569: `dnssec-keygen(1)` can now create encrypted DNSKEY private keys, so that private key material can be stored encrypted at-rest on disk. Until now, private key material could only be stored in clear-text unencrypted form in files named `Knnnn.+aaa+iiii.private` where `<nnnn>` is the key name, `<aaa>` is the numeric representation of the algorithm, and `<iiii>` is the key identifier. If one needed better security for the key material, the only other alternative was to store the keys in a hardware security module (HSM). Now, as another option, the DNSSEC programs allow storing and using private keys in an encrypted `Knnnn.+aaa+iiii.pem` file alongside the existing `Knnnn.+aaa+iiii.key` and `Knnnn.+aaa+iiii.private` files. In this case, the private key material fields are stored in the encrypted `Knnnn.+aaa+iiii.pem` file instead of the clear-text `Knnnn.+aaa+iiii.private` file, whereas the `Knnnn.+aaa+iiii.private` file continues to store key metadata. This protects the private key material at rest.
- RT1592: The Loop system tests have been updated to use modern DNSSEC algorithms, and RSASHA1 and NSEC3RSASHA1 are no longer used except to test specific cases where their use is required. This is in preparation awaiting the approval of [draft-ietf-dnsop-must-not-sha1](#) when support for DNSSEC signing with RSA + SHA-1 will be dropped. `dnssec-keygen(1)` and `dnssec-keyfromlabel(1)` now print a warning if RSASHA1 or NSEC3RSASHA1 keys are created.
- RT1591: DNSSEC programs such as `dnssec-keygen(1)` and `dnssec-keyfromlabel(1)` now require the DNSKEY algorithm to be specified explicitly using the `-a` command-line argument. There is no longer a default selection of the algorithm. References have been added to the manpage on selecting a suitable algorithm. This is to avoid unexpected surprises when these programs are used in scripts and the default algorithm type has to change.
- RT1579: The `platform.h` header which catered to differences in obsolete POSIX platforms has been removed as the source code tree has been modernized in recent years. This is not a user-visible change.

11.4 Loop 1.99.4

The following are release notes for Loop 1.99.4:

- RT1513: Support for Fedora 40 has been dropped as it has reached EOL upstream.
- RT1524: Support for RHEL 7 has been dropped as it has reached end-of-maintenance support upstream. Compiling Loop packages for RHEL 7 had needed extra dependencies such as a newer C compiler from

`devtoolset-7-gcc` with support for `<stdatomic.h>`, and several code and build farm customizations. These have now been dropped.

- RT1543: Several files missing from the dist tarball (source code tarball) were added to `EXTRA_DIST`. These are not user-visible as they're not packaged in the binary packages.
- RT1544: A unittest suite was added for the database versioning functionality (`dbversion`).
- RT1550: A separate memory context is now used for OpenSSL 3 library's memory allocations. This allows the user to do accounting of OpenSSL 3 library's memory allocations.
- RT1551: A benchmark program called `benchmark-zone-queries` was added. This program is not packaged in the binary packages currently.
- RT1557: The task manager (`taskmgr`) was refactored to use atomic operations and remove mutex locking in several places.
- RT1558: Loop's worker threads (part of the task manager) have been refactored to use event loops instead of condition variables. The user visible aspect of this would be that in backtraces, worker threads would no longer be seen blocked on `pthread_cond_wait()` waiting for work. Instead they would be seen blocked on `epoll_wait()` (Linux) or `kevent()` (FreeBSD).
- RT1560: Some unnecessary arrays (proportional to the `<maximum-sockets>` value) and mutex arrays (and associated locking) were removed from the socket manager (`socketmgr`). This was possible due to the use of `struct epoll_event->data.ptr` and `struct kevent->udata` fields available as part of `epoll` and `kevent` respectively. Now, the `<maximum-sockets>` value as provided to `named -S` is simply a limit and does not use any additional resources. It could become an advisory limit in the future.
- RT1562: A member of the socket structure was not initialized leading to a single Valgrind memcheck report which was fixed. There are no more Valgrind memcheck reports generated for **named** during any of our unit and system tests currently.
- RT1515: The object attribute's value provided to `dnssec-keyfromlabel -l` is now properly URI percent-encoded per [RFC 7512](#) and [RFC 3986](#).
- RT1564: The PIN command-line argument for DNSSEC programs has been removed. A PIN can now be provided as part of the PKCS#11 URI itself (see [RFC 7512](#) for the `pin-source` and `pin-value` attributes).
- RT1516: A PKCS#11 HOWTO has been added to the Loop User Manual.
- RT1565: Release notes are now available in the Loop User Manual.

RT1560 and RT1558 involve significant changes to the `socketmgr` and `taskmgr` respectively. Other changes were made to the tree that are not ready for public release.

11.5 Loop 1.99.3

The following are release notes for Loop 1.99.3:

- RT1488: Some textual changes to the *arpaname* manpage were reverted.
- RT1502: OpenSSL and Kerberos library detection was improved. While this may not appear to be a user-visible change, it improves the linker flags that are used.
- RT1500: Support for Red Hat Enterprise Linux 10 was added. Loop RPM packages for RHEL 10 are now available for *x86_64* and *aarch64* platforms.
- RT1503: Support for OpenSSL library versions older than OpenSSL 3 has been dropped. All of Loop's supported platforms have the OpenSSL 3 library.
- RT1503: Usage of the libcrypto library API has been updated to use current OpenSSL 3 APIs, and all deprecated API usage has been removed.
- RT1503: PKCS#11 support has been updated to use the [pkcs11-provider](#) OpenSSL3 provider. The older OpenSSL engine support has been dropped completely. (Examples of PKCS#11 usage for DNSSEC will be documented soon.)
- RT1503: DNSSEC private keys of all the supported DNSKEY algorithms including *RSASHA256*, *RSASHA512*, *ECDSAP256SHA256*, *ECDSAP384SHA384*, *ED25519*, and *ED448* can now be used from PKCS#11 accessible HSMs. All of these algorithms are tested in our automated system tests suite using the [pkcs11-provider](#) OpenSSL3 provider, and the [SoftHSMv2](#) PKCS#11 provider.
- RT1507: *dnssec-keygen(1)* can now be used to directly generate keys on PKCS#11 accessible HSMs for all the supported DNSKEY algorithms including *RSASHA256*, *RSASHA512*, *ECDSAP256SHA256*, *ECDSAP384SHA384*, *ED25519*, and *ED448*. The *-l* argument was added for it. The method of importing the public key of an existing keypair on the HSM using *dnssec-keyfromlabel(1)* can also be used.
- RT1504: Support for the *RSASHA1* DNSKEY algorithm on the OS platform running Loop is checked before use. Some operating systems such as Fedora 41 (and above) and Red Hat Enterprise Linux 10 disable support for *RSASHA1* in their default configuration. Support is checked by signing and verifying some data using the algorithm to see if it succeeds --- the algorithm is disabled (similar to the *disable-algorithms* config option of *named.conf(5)*) if the check fails.
- RT1505: The *named(8)* nameserver now explicitly disables *RSAMD5*, *DSA*, *NSEC3DSA*, and *ECCGOST* in its builtin configuration's *disable-algorithms* option. It also explicitly disables *GOST* in its builtin configuration's *disable-ds-digests* option.
- RT1511: Porting to FreeBSD has been completed and our unit tests and system tests suite pass on it. We expect to publish packages for FreeBSD soon.

11.6 Loop version numbering scheme

Loop version numbers have the grammar <MAJOR>.<MINOR>.<PATCH>.<COMMIT-TIMESTAMP>.<COMMIT-HASH>. The *MAJOR* and *MINOR* version numbers together represent a source code branch of Loop (see [Loop branches](#)).

- The *MAJOR* version number is incremented when configuration options, API, and behavior of features change compared to the existing version. Switching to a new *MAJOR* version may require modifying existing config files to make them compatible with the new version.
- The *MINOR* version number is incremented when new configuration options, API, and features are introduced that are compatible with existing configuration options. Switching to a new *MINOR* version will not require modifying existing config files to make them compatible with the new version.
- The *PATCH* version number is incremented when only bugs have been fixed in a new version. Switching to a new *PATCH* version will not require modifying existing config files to make them compatible with the new version.
- The *COMMIT-TIMESTAMP* field is auto-generated and contains the UTC timestamp of the source code commit from which the Loop release was made. The timestamp value is formatted as YYYYMMDDHHMMSS, as the output of:

```
date +%Y%m%d%H%M%S
```

- The *COMMIT-HASH* field is auto-generated and contains the abbreviated commit hash of the source code commit from which the Loop release was made. The hash value is formatted as the output of:

```
git log -n1 --reverse --pretty=%h
```

For example,

- If you're upgrading from version 1.2.1 to version 1.4.0, Loop's config files should not require any changes. You may also check for new features that have become available in the 1.4 branch.
- If you're upgrading from version 1.2.1 to version 2.0.0, it is possible that some of the contents of your existing config files may need changes. You may also check for new features that have become available in the 2.0 branch.
- If you're upgrading from version 1.2.1 to version 1.2.4, Loop's config files should not require any changes. The newer version only contains bugfixes.

Note

During a major version's release series, features and/or programs scheduled for removal in the next major release may be marked as deprecated. They will however still be supported until the end-of-life of that major release.

11.6.1 Stable and development versions

Even-numbered *minor* versions indicate stable branch releases, whereas odd-numbered *minor* versions indicate development branch releases. For example,

- 1.2.0 is a stable branch release,
- 1.3.0 is a development branch release,
- 1.99.0 is a development branch release,
- 2.0.3 is a stable branch release, and
- 2.1.1 is a development branch release.

Warning

Development branch releases should not be used in production as their features and interfaces may change. Development branch releases may not work properly, may have unexpected behaviors, may crash, etc.

11.7 Loop branches

The following is information on current Loop branches.

Branch	Type	First release date	End-of-life date
1.99	Development	2024-12-10	To be announced
2.0	Stable	To be announced	To be announced

Development branches have no planned end-of-life. Typically, development on such branches is stopped when a new *MINOR*+1 or *MAJOR*+1 stable branch is created off it.

11.8 History of Loop

Although the official beginning of the Domain Name System occurred in 1984 with the publication of RFC 920, the core of the new system was described in 1983 in RFCs 882 and 883. From 1984 to 1987, the ARPAnet (the precursor to today's Internet) became a testbed of experimentation for developing the new naming/addressing scheme in a rapidly expanding, operational network environment. New RFCs were written and published in 1987 that modified the original documents to incorporate improvements based on the working model: RFC 1034 (*Domain Names-Concepts and Facilities*), and RFC 1035 (*Domain Names-Implementation and Specification*) were published and became the standards upon which all DNS implementations are built.

The first working domain name server, called *Jeeves*, was written in 1983--84 by Paul Mockapetris for operation on DEC Tops-20 machines located at the University of Southern California's Information Sciences Institute (USC-ISI) and SRI International's

Network Information Center (SRI-NIC). A DNS server for Unix machines, the Berkeley Internet Name Domain (BIND) package, was written soon after by a group of graduate students at the University of California at Berkeley under a grant from the US Defense Advanced Research Projects Administration (DARPA).

Versions of BIND through 4.8.3 were maintained by the Computer Systems Research Group (CSRG) at UC Berkeley. Douglas Terry, Mark Painter, David Riggle and Songnian Zhou made up the initial BIND project team. After that, additional work on the software package was done by Ralph Campbell. Kevin Dunlap, a Digital Equipment Corporation employee on loan to the CSRG, worked on BIND for 2 years, from 1985 to 1987. Many other people also contributed to BIND development during that time: Doug Kingston, Craig Partridge, Smoot Carl-Mitchell, Mike Muuss, Jim Bloom and Mike Schwartz. BIND maintenance was subsequently handled by Mike Karels and Øivind Kure.

BIND versions 4.9 and 4.9.1 were released by Digital Equipment Corporation (now Compaq Computer Corporation). Paul Vixie, then a DEC employee, became BIND's primary caretaker. He was assisted by Phil Almquist, Robert Elz, Alan Barrett, Paul Albitz, Bryan Beecher, Andrew Partan, Andy Cherenon, Tom Limoncelli, Berthold Paffrath, Fuat Baran, Anant Kumar, Art Harkin, Win Treese, Don Lewis, Christophe Wolfhugel, and others.

In 1994, BIND version 4.9.2 was sponsored by Vixie Enterprises. Paul Vixie became BIND's principal architect/programmer.

BIND versions from 4.9.3 onward have been developed and maintained by the Internet Systems Consortium (ISC) and its predecessor, the Internet Software Consortium, with support being provided by ISC's sponsors. As co-architects/programmers, Bob Halley and Paul Vixie released the first production-ready version of BIND version 8 in May 1997. BIND version 9 was released in September 2000 and is a major rewrite of nearly all aspects of the underlying BIND architecture. A BIND 10 project was started by ISC in 2009 to build next-generation DNS and DHCP implementations, but it failed to reach its goals and was abandoned in 2014 after 5 years of development. ISC returned to focusing on BIND 9 from which additional releases continue to be made. A subscription branch of BIND 9 with some proprietary features was also started by ISC during this time that is available only to paying customers. With the release of BIND 9.11.0, BIND 9 underwent a license change from the ISC license to the Mozilla Public License version 2. The BIND 9 codebase continues to be developed by ISC.

Being what may be considered as a reference implementation of DNS, BIND 9 implemented many DNS related features. The BIND 9 codebase was developed starting in August 1998 and was well-designed with suitable use of encapsulation and abstraction. But due to decades of several different sets of internal and external developers extending and patching the code, its codebase became highly complex and unwieldy. BIND 9's competitors have caught up to it in the features that matter, and its performance compares very poorly to what is routinely achieved by its competitors. A major recent complaint of DNS operators is the high number of security vulnerabilities that are regularly found and fixed in BIND 9. This may be attributed to the breath of features and complexity which makes it difficult for BIND 9 to avoid vulnerabilities. The development team at ISC are efficient at fixing reported vulnerabilities quickly, and its

releases are generally well-supported by ISC's staff.

Loop was started by a former BIND 9 developer with the goal of reducing the number of features in this codebase so that it still serves a large percentage of typical DNS operational use-cases, but with greatly reduced code complexity, improved quality of code and maintainability. Loop aims to serve some additional use-cases where BIND 9 is not applied currently. Loop forked from the BIND 9.10.8 release and incorporates some code from the BIND 9.11.0a2 release and some bits from BIND 10. While externally it still resembles BIND 9 with its similar configuration language and programs, its code has undergone considerable changes and continues to evolve at a high rate.

CHAPTER TWELVE

LICENSE

Copyright (C) 2024-2025 Banu Systems Private Limited. All rights reserved.

This product is not open source software. Permission is not granted to redistribute this product.

A proprietary software license is used as an overall license for distributing binaries, documentation, and other works of this product.

THE SOFTWARE IS PROVIDED "AS IS" AND BANU DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL BANU BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Portions of this product are covered by one or more of the following original copyright and license notices. This product cannot re-license the original unmodified effects and such portions of the product are covered by their original licenses listed below.

Copyright (C) 1996-2018 Internet Systems Consortium, Inc. ("ISC")

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND ISC DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (C) 1996-2001 Nominum, Inc. Copyright (C) 2000, 2001, 2004-2015 Nominum, Inc.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND NOMINUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL NOMINUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (C) 1995-2000 by Network Associates, Inc.

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND ISC AND NETWORK ASSOCIATES DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (C) 2002 Stichting NLnet, Netherlands, stichting@nlnet.nl.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND STICHTING NLNET DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL STICHTING NLNET BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (c) 1987, 1990, 1993, 1994 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) The Internet Society 2005. This version of this module is part of RFC 4178; see the RFC itself for full legal notices.

(The above copyright notice is per RFC 3978 5.6 (a), q.v.)

Copyright (c) 2004 Masarykova universita (Masaryk University, Brno, Czech Republic)
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1997 - 2003 Kungliga Tekniska Hogskolan (Royal Institute of Technology, Stockholm, Sweden). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1998 Doug Rabson All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright ((c)) 2002, Rice University All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Rice University (RICE) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by RICE and the contributors on an "as is" basis, without any representations or warranties of any kind, express or implied including, but not limited to, representations or warranties of non-infringement, merchantability or fitness for a particular purpose. In no event shall RICE or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Copyright (c) 1993 by Digital Equipment Corporation.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies, and that the name of Digital Equipment Corporation not be used in advertising or publicity pertaining to distribution of the document or software without specific, written prior permission.

THE SOFTWARE IS PROVIDED "AS IS" AND DIGITAL EQUIPMENT CORP. DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO

EVENT SHALL DIGITAL EQUIPMENT CORPORATION BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright 2000 Aaron D. Gifford. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR(S) AND CONTRIBUTOR(S) ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR(S) OR CONTRIBUTOR(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1998 Doug Rabson. Copyright (c) 2001 Jake Burkholder. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS

FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1999-2000 by Nortel Networks Corporation

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND NORTEL NETWORKS DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL NORTEL NETWORKS BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF

CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (c) 2000-2002 Japan Network Information Center. All rights reserved.

By using this file, you agree to the terms and conditions set forth bellow.

LICENSE TERMS AND CONDITIONS

The following License Terms and Conditions apply, unless a different license is obtained from Japan Network Information Center ("JPNIC"), a Japanese association, Kokusai-Kougyou-Kanda Bldg 6F, 2-3-4 Uchi-Kanda, Chiyoda-ku, Tokyo 101-0047, Japan.

1. Use, Modification and Redistribution (including distribution of any modified or derived work) in source and/or binary forms is permitted under this License Terms and Conditions.
2. Redistribution of source code must retain the copyright notices as they appear in each source code file, this License Terms and Conditions.
3. Redistribution in binary form must reproduce the Copyright Notice, this License Terms and Conditions, in the documentation and/or other materials provided with the distribution. For the purposes of binary distribution the "Copyright Notice" refers to the following language: "Copyright (c) 2000-2002 Japan Network Information Center. All rights reserved."
4. The name of JPNIC may not be used to endorse or promote products derived from this Software without specific prior written approval of JPNIC.
5. Disclaimer/Limitation of Liability: THIS SOFTWARE IS PROVIDED BY JPNIC "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JPNIC BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Copyright (C) 2004 Nominet, Ltd.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND NOMINET DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Portions Copyright RSA Security Inc.

License to copy and use this software is granted provided that it is identified as "RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)" in all material mentioning or referencing this software.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)" in all material mentioning or referencing the derived work.

RSA Security Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

Copyright (c) 1996, David Mazieres <dm@uun.org> Copyright (c) 2008, Damien Miller <djm@openbsd.org> Copyright (c) 2013, Markus Friedl <markus@openbsd.org> Copyright (c) 2014, Theo de Raadt <deraadt@openbsd.org>

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (c) 2000-2001 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
-

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.OpenSSL.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact licensing@OpenSSL.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.OpenSSL.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1995, 1997, 1998 The NetBSD Foundation, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DATA AND PRIVACY

Loop implements---as part of its software features---support for logging various types of activity for the administrator to use. For example, DNS queries may be logged using the `querylog` option of `named.conf(5)`.

- Logging may be performed on behalf of the administrator on the machine where the Loop software is installed.
- Log messages may contain personal data about a user such as, but not limited to, IP addresses, MAC addresses, hostnames, DNS names, TSIG key names, timestamps, etc.

We note that:

- Loop does not share user data or logs with Banu Systems Private Limited, unless the administrator explicitly configures it to do so.
- Loop does not share user data or logs with other companies, organizations, or persons, unless the administrator explicitly configures it to do so.

Symbols

- +aaflag
 - command line option, 75
- +additional
 - command line option, 75
- +adflag
 - command line option, 75
- +all
 - command line option, 70, 75
- +answer
 - command line option, 76
- +authority
 - command line option, 76
- +besteffort
 - command line option, 76
- +bufsize
 - command line option, 76
- +cdflag
 - command line option, 69, 76
- +class
 - command line option, 69, 76
- +cmd
 - command line option, 76
- +comments
 - command line option, 70, 76
- +cookie
 - command line option, 76
- +crypto
 - command line option, 70, 77
- +dnssec
 - command line option, 71, 77
- +domain
 - command line option, 77
- +edns
 - command line option, 77
- +ednsflags
 - command line option, 77
- +ednsnegotiation
 - command line option, 77
- +ednsopt
 - command line option, 77
- +expire
 - command line option, 78
- +fail
 - command line option, 78
- +identify
 - command line option, 78
- +ignore
 - command line option, 78
- +keepopen
 - command line option, 78
- +mtrace
 - command line option, 69
- +multiline
 - command line option, 71, 78
- +ndots
 - command line option, 78
- +noaaflag
 - command line option, 75
- +noadditional
 - command line option, 75
- +noadflag
 - command line option, 75
- +noall
 - command line option, 71, 75
- +noanswer
 - command line option, 76
- +noauthority
 - command line option, 76
- +nobesteffort
 - command line option, 76
- +nocdflag
 - command line option, 69, 76
- +noclass
 - command line option, 69, 76
- +nocmd

command line option,76	command line option,71
+nocomments	+norrcomments
command line option,70,76	command line option,70,80
+nocookie	+nortrace
command line option,77	command line option,69
+nocrypto	+nosearch
command line option,70,77	command line option,80
+nodnssec	+noshort
command line option,71,77	command line option,70,80
+noedns	+noshowsearch
command line option,77	command line option,80
+noednsflags	+nosit
command line option,77	command line option,80
+noednsnegotiation	+nosplit
command line option,77	command line option,70,80
+noednsopt	+nostats
command line option,77	command line option,80
+noexpire	+nosubnet
command line option,78	command line option,81
+nofail	+notcp
command line option,78	command line option,81
+noidentify	+notrace
command line option,78	command line option,81
+noignore	+notrust
command line option,78	command line option,70
+nokeepopen	+nottl
command line option,78	command line option,69
+nomtrace	+nottlid
command line option,69	command line option,81
+nomultiline	+novc
command line option,71,78	command line option,82
+nonsid	+novtrace
command line option,79	command line option,70
+nonssearch	+nsid
command line option,79	command line option,78
+noonesoa	+nssearch
command line option,79	command line option,79
+noopcode	+onesoa
command line option,79	command line option,79
+noqr	+opcode
command line option,79	command line option,79
+noquestion	+qr
command line option,79	command line option,79
+nordflag	+question
command line option,79	command line option,79
+norecurse	+rdflag
command line option,79	command line option,79
+noroot	+recurse

command line option, 79	-6	
+retry		command line option, 19, 68, 73, 83
command line option, 79		
+root	-A	
command line option, 71		command line option, 32, 42, 48, 53, 57, 63
+rrcomments		
command line option, 70, 80	-C	
+rtrace		command line option, 40, 41, 59, 84, 105
command line option, 69		
+search	-D	
command line option, 80		command line option, 20, 39, 44, 47, 53, 57, 59, 87, 94, 104
+short	-E	
command line option, 70, 80		command line option, 50, 55, 56, 59
+showsearch		
command line option, 80	-F	
+sit		command line option, 20
command line option, 80	-G	
+split		command line option, 46, 50
command line option, 70, 80	-H	
+stats		command line option, 63
command line option, 80	-I	
+subnet		command line option, 48, 53, 57
command line option, 81	-J	
+tcp		command line option, 37
command line option, 81	-K	
+time		command line option, 41, 43, 46, 51, 55, 56, 59
command line option, 81	-L	
+trace		command line option, 38, 44, 46, 51, 56, 61, 87, 105
command line option, 81	-M	
+tries		command line option, 38, 59
command line option, 81	-N	
+trust		command line option, 61, 84
command line option, 70	-O	
+ttl		command line option, 61
command line option, 69	-P	
+ttlid		command line option, 40, 44, 47, 52, 57, 61, 87, 105
command line option, 81	-Q	
+vc		command line option, 62, 95
command line option, 82	-R	
+vtrace		command line option, 48, 53, 55, 57, 62, 84
command line option, 70	-S	
@<server>		command line option, 21, 38, 46, 52, 58, 62, 95
command line option, 67, 72		
-3		
command line option, 63		
-4		
command line option, 19, 68, 73, 83		

-T	84, 87, 95
command line option, 21, 39–41, 47, 52, 62, 85, 88, 95	-m command line option, 38, 104
-U	-n
command line option, 22	command line option, 20, 38, 46, 50, 61, 95
-V	-o
command line option, 22, 25, 33, 35, 36, 39, 40, 42, 44, 47, 52, 55, 57, 63, 65, 68, 74, 85, 88, 96, 105	command line option, 38, 40, 61, 65
-W	-p
command line option, 39, 85	command line option, 20, 25, 33, 36, 40, 46, 51, 58, 67, 73, 87, 95, 103
-X	-q
command line option, 60	command line option, 25, 34, 37, 51, 67, 73, 95, 105
-a	-r
command line option, 32, 34, 41, 45, 49, 59, 67, 83, 94, 103	command line option, 38, 55, 84, 87, 105
-b	-s
command line option, 24, 32, 49, 67, 73, 94, 104	command line option, 21, 24, 33, 34, 38, 42, 52, 60, 85, 95, 103
-c	-t
command line option, 20, 24, 32, 37, 42, 46, 50, 59, 65, 67, 73, 83, 94, 105	command line option, 21, 33, 36, 39, 47, 52, 63, 67, 74, 85, 87, 95, 104
-d	-u
command line option, 20, 37, 59, 68, 84, 87, 94, 103	command line option, 21, 33, 40, 58, 63, 74, 88, 96
-e	-v
command line option, 50, 60, 95, 104	command line option, 25, 33, 42, 44, 47, 52, 55, 57, 63, 65, 85, 88, 96
-f	-w
command line option, 20, 41, 43, 46, 50, 55, 56, 60, 73, 95, 104	command line option, 39, 85
-g	-x
command line option, 20, 50, 59	command line option, 22, 36, 63, 65, 68, 74, 96, 103
-h	-y
command line option, 20, 24, 33, 34, 36, 37, 40, 42, 44, 46, 52, 55, 56, 63, 65, 68, 73, 84, 87, 95, 104	command line option, 25, 47, 74, 88, 96, 104
-i	-z
command line option, 37, 48, 53, 57, 60, 67, 73, 84, 87, 104	command line option, 35, 36, 63, 65
-j	<algorithm>
command line option, 36, 37, 60	command line option, 31
-k	<class>
command line option, 24, 33, 34, 38, 59, 73, 87	command line option, 68, 75
-l	<command>
command line option, 38, 45, 51,	command line option, 25
	<domain>

- command line option, 31
- <filename>
 - command line option, 36, 39
- <ip-address>
 - command line option, 31
- <iterations>
 - command line option, 31
- <name>
 - command line option, 47, 52, 68, 75
- <salt>
 - command line option, 31
- <type>
 - command line option, 68, 75
- <zonefile>
 - command line option, 64, 66
- <zonename>
 - command line option, 39
- [<key>
 - command line option, 64

A

- addzone
 - command line option, 25
- allow-new-zones
 - command line option, 137
- allow-notify
 - command line option, 152
- allow-query
 - command line option, 152
- allow-query-cache
 - command line option, 137
- allow-query-cache-on
 - command line option, 137
- allow-query-on
 - command line option, 152
- allow-recursion
 - command line option, 138
- allow-recursion-on
 - command line option, 138
- allow-transfer
 - command line option, 152
- allow-update
 - command line option, 153
- allow-update-forwarding
 - command line option, 153
- also-notify
 - command line option, 153

- alt-transfer-source
 - command line option, 153
- alt-transfer-source-v6
 - command line option, 154
- answer
 - command line option, 91
- attach-cache
 - command line option, 138
- auth-nxdomain
 - command line option, 139
- auto-dnssec
 - command line option, 154
- automatic-interface-scan
 - command line option, 129
- avoid-v4-udp-ports
 - command line option, 129
- avoid-v6-udp-ports
 - command line option, 129

B

- block
 - command line option, 129
- bogus
 - command line option, 166

C

- cache-file
 - command line option, 139
- check-dup-records
 - command line option, 155
- check-integrity
 - command line option, 155
- check-mx
 - command line option, 155
- check-mx-cname
 - command line option, 155
- check-names
 - command line option, 139
- check-sibling
 - command line option, 155
- check-spf
 - command line option, 155
- check-srv-cname
 - command line option, 155
- check-wildcard
 - command line option, 156
- class
 - command line option, 89

clients-per-query
 command line option, 140

command line option
 +aaflag, 75
 +additional, 75
 +adflag, 75
 +all, 70, 75
 +answer, 76
 +authority, 76
 +besteffort, 76
 +bufsize, 76
 +cdflag, 69, 76
 +class, 69, 76
 +cmd, 76
 +comments, 70, 76
 +cookie, 76
 +crypto, 70, 77
 +dnssec, 71, 77
 +domain, 77
 +edns, 77
 +ednsflags, 77
 +ednsnegotiation, 77
 +ednsopt, 77
 +expire, 78
 +fail, 78
 +identify, 78
 +ignore, 78
 +keepopen, 78
 +mtrace, 69
 +multiline, 71, 78
 +ndots, 78
 +noaaflag, 75
 +noadditional, 75
 +noadflag, 75
 +noall, 71, 75
 +noanswer, 76
 +noauthority, 76
 +nobesteffort, 76
 +nocdflag, 69, 76
 +noclass, 69, 76
 +nocmd, 76
 +nocomments, 70, 76
 +nocookie, 77
 +nocrypto, 70, 77
 +nodnssec, 71, 77
 +noedns, 77
 +noednsflags, 77
 +noednsnegotiation, 77
 +noednsopt, 77
 +noexpire, 78
 +nofail, 78
 +noidentify, 78
 +noignore, 78
 +nokeepopen, 78
 +nomtrace, 69
 +nomultiline, 71, 78
 +nonsid, 79
 +nonssearch, 79
 +noonesoa, 79
 +noopcode, 79
 +noqr, 79
 +noquestion, 79
 +nordflag, 79
 +norecurse, 79
 +noroot, 71
 +norrcomments, 70, 80
 +nortrace, 69
 +nosearch, 80
 +noshort, 70, 80
 +noshowsearch, 80
 +nosit, 80
 +nosplit, 70, 80
 +nostats, 80
 +nosubnet, 81
 +notcp, 81
 +notrace, 81
 +notrust, 70
 +nottl, 69
 +nottlid, 81
 +novc, 82
 +novtrace, 70
 +nsid, 78
 +nssearch, 79
 +onesoa, 79
 +opcode, 79
 +qr, 79
 +question, 79
 +rdflag, 79
 +recurse, 79
 +retry, 79
 +root, 71
 +rrcomments, 70, 80
 +rtrace, 69
 +search, 80
 +short, 70, 80
 +showsearch, 80

- +sit, 80
- +split, 70, 80
- +stats, 80
- +subnet, 81
- +tcp, 81
- +time, 81
- +trace, 81
- +tries, 81
- +trust, 70
- +ttl, 69
- +ttlid, 81
- +vc, 82
- +vtrace, 70
- @<server>, 67, 72
- 3, 63
- 4, 19, 68, 73, 83
- 6, 19, 68, 73, 83
- A, 32, 42, 48, 53, 57, 63
- C, 40, 41, 59, 84, 105
- D, 20, 39, 44, 47, 53, 57, 59, 87, 94, 104
- E, 50, 55, 56, 59
- F, 20
- G, 46, 50
- H, 63
- I, 48, 53, 57
- J, 37
- K, 41, 43, 46, 51, 55, 56, 59
- L, 38, 44, 46, 51, 56, 61, 87, 105
- M, 38, 59
- N, 61, 84
- O, 61
- P, 40, 44, 47, 52, 57, 61, 87, 105
- Q, 62, 95
- R, 48, 53, 55, 57, 62, 84
- S, 21, 38, 46, 52, 58, 62, 95
- T, 21, 39–41, 47, 52, 62, 85, 88, 95
- U, 22
- V, 22, 25, 33, 35, 36, 39, 40, 42, 44, 47, 52, 55, 57, 63, 65, 68, 74, 85, 88, 96, 105
- W, 39, 85
- X, 60
- a, 32, 34, 41, 45, 49, 59, 67, 83, 94, 103
- b, 24, 32, 49, 67, 73, 94, 104
- c, 20, 24, 32, 37, 42, 46, 50, 59, 65, 67, 73, 83, 94, 105
- d, 20, 37, 59, 68, 84, 87, 94, 103
- e, 50, 60, 95, 104
- f, 20, 41, 43, 46, 50, 55, 56, 60, 73, 95, 104
- g, 20, 50, 59
- h, 20, 24, 33, 34, 36, 37, 40, 42, 44, 46, 52, 55, 56, 63, 65, 68, 73, 84, 87, 95, 104
- i, 37, 48, 53, 57, 60, 67, 73, 84, 87, 104
- j, 36, 37, 60
- k, 24, 33, 34, 38, 59, 73, 87
- l, 38, 45, 51, 84, 87, 95
- m, 38, 104
- n, 20, 38, 46, 50, 61, 95
- o, 38, 40, 61, 65
- p, 20, 25, 33, 36, 40, 46, 51, 58, 67, 73, 87, 95, 103
- q, 25, 34, 37, 51, 67, 73, 95, 105
- r, 38, 55, 84, 87, 105
- s, 21, 24, 33, 34, 38, 42, 52, 60, 85, 95, 103
- t, 21, 33, 36, 39, 47, 52, 63, 67, 74, 85, 87, 95, 104
- u, 21, 33, 40, 58, 63, 74, 88, 96
- v, 25, 33, 42, 44, 47, 52, 55, 57, 63, 65, 85, 88, 96
- w, 39, 85
- x, 22, 36, 63, 65, 68, 74, 96, 103
- y, 25, 47, 74, 88, 96, 104
- z, 35, 36, 63, 65
- <algorithm>, 31
- <class>, 68, 75
- <command>, 25
- <domain>, 31
- <filename>, 36, 39
- <ip-address>, 31
- <iterations>, 31
- <name>, 47, 52, 68, 75
- <salt>, 31
- <type>, 68, 75
- <zonefile>, 64, 66
- <zonename>, 39
- [<key>, 64
- addzone, 25
- allow-new-zones, 137
- allow-notify, 152
- allow-query, 152
- allow-query-cache, 137
- allow-query-cache-on, 137
- allow-query-on, 152

allow-recursion, 138
 allow-recursion-on, 138
 allow-transfer, 152
 allow-update, 153
 allow-update-forwarding, 153
 also-notify, 153
 alt-transfer-source, 153
 alt-transfer-source-v6, 154
 answer, 91
 attach-cache, 138
 auth-nxdomain, 139
 auto-dnssec, 154
 automatic-interface-scan, 129
 avoid-v4-udp-ports, 129
 avoid-v6-udp-ports, 129
 block, 129
 bogus, 166
 cache-file, 139
 check-dup-records, 155
 check-integrity, 155
 check-mx, 155
 check-mx-cname, 155
 check-names, 139
 check-sibling, 155
 check-spf, 155
 check-srv-cname, 155
 check-wildcard, 156
 class, 89
 clients-per-query, 140
 cookie-algorithm, 135
 cookie-secret, 135
 coresize, 130
 datasize, 130
 debug, 91
 delzone, 26
 deny-answer-addresses, 140
 deny-answer-aliases, 140
 dialup, 156
 directory, 130
 disable-algorithms, 140
 disable-ds-digests, 140
 disable-empty-zone, 140
 dns64, 141
 dns64-contact, 141
 dns64-server, 142
 dnssec-accept-expired, 132
 dnssec-dnskey-kskonly, 157
 dnssec-enable, 142
 dnssec-keys-file, 129
 dnssec-loadkeys-interval, 157
 dnssec-must-be-secure, 142
 dnssec-secure-to-insecure, 157
 dnssec-update-mode, 157
 dnssec-validation, 142
 dscp, 130
 dual-stack-servers, 142
 dump-file, 130
 dumpdb, 26
 edns, 166
 edns-udp-size, 143, 166
 empty-contact, 143
 empty-server, 143
 empty-zones-enable, 144
 fetch-quota-params, 144
 fetches-per-server, 144
 fetches-per-zone, 145
 file, 114
 files, 130
 flush, 26
 flush-zones-on-shutdown, 131
 flushname, 26
 flushtree, 26
 forward, 158
 forwarders, 158
 freeze, 26
 gsstsig, 89
 halt, 26
 heartbeat-interval, 131
 help, 91
 inline-signing, 158
 interface-interval, 131
 ixfr-from-differences, 145
 key, 89
 key-directory, 158
 keys, 166
 listen-on, 131
 listen-on-v6, 131
 loadkeys, 27
 local, 89
 managed-keys-directory, 131
 match-mapped-addresses, 132
 max-cache-size, 146
 max-cache-ttl, 146
 max-clients-per-query, 146
 max-journal-size, 159

max-ncache-ttl, 147
 max-records, 159
 max-recursion-depth, 147
 max-recursion-queries, 147
 max-refresh-time, 159
 max-retry-time, 159
 max-rsa-exponent-size, 132
 max-transfer-idle-in, 159
 max-transfer-idle-out, 159
 max-transfer-time-in, 159
 max-transfer-time-out, 160
 max-udp-size, 147, 167
 max-zone-ttl, 160
 min-refresh-time, 160
 min-retry-time, 160
 minimal-responses, 147
 multi-master, 160
 no-case-compress, 147
 nocookie-udp-size, 148
 notify, 27, 160
 notify-delay, 161
 notify-source, 161, 167
 notify-source-v6, 161, 167
 notify-to-soa, 161
 notrace, 27
 nsec3-test-zone, 161
 null, 115
 oldgsstsig, 90
 pid-file, 132
 port, 132
 preferred-glue, 148
 prefetch, 148
 prereq, 90
 print-category, 115
 print-severity, 115
 print-time, 115
 provide-ixfr, 149, 167
 query-source, 149, 167
 query-source-v6, 149, 168
 querylog, 27, 133
 rate-limit, 149
 realm, 90
 reconfig, 27
 recursing, 27
 recursing-file, 133
 recursion, 150
 recursive-clients, 133
 refresh, 27
 reload, 27
 request-cookie, 151, 168
 request-ixfr, 161, 168
 request-nsid, 150, 168
 reserved-sockets, 133
 resolver-query-timeout, 151
 response-policy, 151
 retransfer, 27
 root-delegation-only, 151
 rrset-order, 152
 scan, 28
 secroots, 28
 secroots-file, 133
 send, 90
 serial-query-rate, 133
 serial-update-method, 162
 server, 89
 server-id, 134
 session-keyalg, 134
 session-keyfile, 134
 session-keyname, 134
 severity, 115
 show, 90
 sig-signing-nodes, 162
 sig-signing-signatures, 162
 sig-signing-type, 162
 sig-validity-interval, 163
 sign, 28
 signing, 28
 stacksize, 135
 statistics-file, 135
 stats, 29
 status, 29
 stderr, 115
 stop, 29
 sync, 29
 syslog, 114
 tcp-clients, 135
 tcp-listen-queue, 135
 tcp-only, 168
 thaw, 29
 tkey-dhkey, 135
 tkey-domain, 136
 tkey-gssapi-credential, 136
 tkey-gssapi-keytab, 136
 trace, 29
 transfer-source, 163, 168
 transfer-source-v6, 163, 168

- transfers, 168
- transfers-in, 136
- transfers-out, 136
- transfers-per-ns, 137
- try-tcp-refresh, 164
- tsig-delete, 30
- tsig-list, 30
- ttl, 89
- update, 90
- update-check-ksk, 164
- use-alt-transfer-source, 164
- use-v4-udp-ports, 137
- use-v6-udp-ports, 137
- validation, 30
- version, 91, 137
- zero-no-soa-ttl, 164
- zero-no-soa-ttl-cache, 152
- zone, 89
- zone-statistics, 164
- zonestatus, 30

- cookie-algorithm
 - command line option, 135
- cookie-secret
 - command line option, 135
- coresize
 - command line option, 130

D

- datasize
 - command line option, 130
- debug
 - command line option, 91
- delzone
 - command line option, 26
- deny-answer-addresses
 - command line option, 140
- deny-answer-aliases
 - command line option, 140
- dialup
 - command line option, 156
- directory
 - command line option, 130
- disable-algorithms
 - command line option, 140
- disable-ds-digests
 - command line option, 140
- disable-empty-zone
 - command line option, 140

- dns64
 - command line option, 141
- dns64-contact
 - command line option, 141
- dns64-server
 - command line option, 142
- dnssec-accept-expired
 - command line option, 132
- dnssec-dnskey-kskonly
 - command line option, 157
- dnssec-enable
 - command line option, 142
- dnssec-keys-file
 - command line option, 129
- dnssec-loadkeys-interval
 - command line option, 157
- dnssec-must-be-secure
 - command line option, 142
- dnssec-secure-to-insecure
 - command line option, 157
- dnssec-update-mode
 - command line option, 157
- dnssec-validation
 - command line option, 142
- dscp
 - command line option, 130
- dual-stack-servers
 - command line option, 142
- dump-file
 - command line option, 130
- dumpdb
 - command line option, 26

E

- edns
 - command line option, 166
- edns-udp-size
 - command line option, 143, 166
- empty-contact
 - command line option, 143
- empty-server
 - command line option, 143
- empty-zones-enable
 - command line option, 144

F

- fetch-quota-params
 - command line option, 144

fetches-per-server
 command line option, [144](#)
 fetches-per-zone
 command line option, [145](#)
 file
 command line option, [114](#)
 files
 command line option, [130](#)
 flush
 command line option, [26](#)
 flush-zones-on-shutdown
 command line option, [131](#)
 flushname
 command line option, [26](#)
 flushtree
 command line option, [26](#)
 forward
 command line option, [158](#)
 forwarders
 command line option, [158](#)
 freeze
 command line option, [26](#)

G

gsstsig
 command line option, [89](#)

H

halt
 command line option, [26](#)
 heartbeat-interval
 command line option, [131](#)
 help
 command line option, [91](#)

I

inline-signing
 command line option, [158](#)
 interface-interval
 command line option, [131](#)
 ixfr-from-differences
 command line option, [145](#)

K

key
 command line option, [89](#)
 key-directory
 command line option, [158](#)
 keys

 command line option, [166](#)

L

listen-on
 command line option, [131](#)
 listen-on-v6
 command line option, [131](#)
 loadkeys
 command line option, [27](#)
 local
 command line option, [89](#)

M

managed-keys-directory
 command line option, [131](#)
 match-mapped-addresses
 command line option, [132](#)
 max-cache-size
 command line option, [146](#)
 max-cache-ttl
 command line option, [146](#)
 max-clients-per-query
 command line option, [146](#)
 max-journal-size
 command line option, [159](#)
 max-ncache-ttl
 command line option, [147](#)
 max-records
 command line option, [159](#)
 max-recursion-depth
 command line option, [147](#)
 max-recursion-queries
 command line option, [147](#)
 max-refresh-time
 command line option, [159](#)
 max-retry-time
 command line option, [159](#)
 max-rsa-exponent-size
 command line option, [132](#)
 max-transfer-idle-in
 command line option, [159](#)
 max-transfer-idle-out
 command line option, [159](#)
 max-transfer-time-in
 command line option, [159](#)
 max-transfer-time-out
 command line option, [160](#)
 max-udp-size

- command line option, [147, 167](#)
- max-zone-ttl
 - command line option, [160](#)
- min-refresh-time
 - command line option, [160](#)
- min-retry-time
 - command line option, [160](#)
- minimal-responses
 - command line option, [147](#)
- multi-master
 - command line option, [160](#)

N

- no-case-compress
 - command line option, [147](#)
- nocookie-udp-size
 - command line option, [148](#)
- notify
 - command line option, [27, 160](#)
- notify-delay
 - command line option, [161](#)
- notify-source
 - command line option, [161, 167](#)
- notify-source-v6
 - command line option, [161, 167](#)
- notify-to-soa
 - command line option, [161](#)
- notrace
 - command line option, [27](#)
- nsec3-test-zone
 - command line option, [161](#)
- null
 - command line option, [115](#)

O

- oldgsstsig
 - command line option, [90](#)

P

- pid-file
 - command line option, [132](#)
- port
 - command line option, [132](#)
- preferred-glue
 - command line option, [148](#)
- prefetch
 - command line option, [148](#)
- prereq

- command line option, [90](#)
- print-category
 - command line option, [115](#)
- print-severity
 - command line option, [115](#)
- print-time
 - command line option, [115](#)
- provide-ixfr
 - command line option, [149, 167](#)

Q

- query-source
 - command line option, [149, 167](#)
- query-source-v6
 - command line option, [149, 168](#)
- querylog
 - command line option, [27, 133](#)

R

- rate-limit
 - command line option, [149](#)
- realm
 - command line option, [90](#)
- reconfig
 - command line option, [27](#)
- recurring
 - command line option, [27](#)
- recurring-file
 - command line option, [133](#)
- recursion
 - command line option, [150](#)
- recursive-clients
 - command line option, [133](#)
- refresh
 - command line option, [27](#)
- reload
 - command line option, [27](#)
- request-cookie
 - command line option, [151, 168](#)
- request-ixfr
 - command line option, [161, 168](#)
- request-nsid
 - command line option, [150, 168](#)
- reserved-sockets
 - command line option, [133](#)
- resolver-query-timeout
 - command line option, [151](#)
- response-policy

- command line option, [151](#)
- retransfer
 - command line option, [27](#)
- RFC
 - RFC 1034, [39](#), [91](#), [118](#), [156](#)
 - RFC 1035, [148](#), [256](#), [257](#)
 - RFC 1123, [139](#)
 - RFC 1886, [73](#), [84](#)
 - RFC 1912, [162](#)
 - RFC 1918, [226](#)
 - RFC 1982, [61](#)
 - RFC 1995, [149](#), [162](#), [167](#), [196](#)
 - RFC 1996, [152](#), [194](#)
 - RFC 2104, [24](#), [201](#)
 - RFC 2136, [86](#), [195](#)
 - RFC 2308, [147](#)
 - RFC 2535, [46](#), [49](#), [51](#), [86](#), [91](#), [204](#)
 - RFC 2539, [51](#)
 - RFC 2845, [49](#), [86](#), [201](#)
 - RFC 2874, [73](#), [84](#)
 - RFC 2930, [49](#), [204](#)
 - RFC 2931, [86](#), [204](#)
 - RFC 3597, [40](#), [74](#), [88](#)
 - RFC 3757, [46](#), [50](#)
 - RFC 3849, [226](#)
 - RFC 3986, [279](#)
 - RFC 4034, [46](#), [49](#), [50](#)
 - RFC 4193, [226](#)
 - RFC 4635, [201](#)
 - RFC 4648, [112](#), [188](#), [276](#)
 - RFC 5001, [79](#), [134](#), [150](#)
 - RFC 5011, [46](#), [47](#), [50](#), [55](#), [67](#), [122](#), [213](#)
 - RFC 5735, [226](#)
 - RFC 5737, [226](#)
 - RFC 5936, [149](#), [162](#), [167](#)
 - RFC 6052, [141](#)
 - RFC 6598, [226](#)
 - RFC 6781, [45](#), [49](#), [50](#), [62](#)
 - RFC 6891, [143](#), [166](#)
 - RFC 7314, [78](#)
 - RFC 7512, [279](#)
 - RFC 7534, [226](#)
 - RFC 7871, [81](#)
 - RFC 7873, [151](#)
 - RFC 821, [139](#)
 - RFC 8375, [226](#)
 - RFC 8945, [201](#)
 - RFC 952, [139](#)

- root-delegation-only
 - command line option, [151](#)
- rrset-order
 - command line option, [152](#)

S

- scan
 - command line option, [28](#)
- secroots
 - command line option, [28](#)
- secroots-file
 - command line option, [133](#)
- send
 - command line option, [90](#)
- serial-query-rate
 - command line option, [133](#)
- serial-update-method
 - command line option, [162](#)
- server
 - command line option, [89](#)
- server-id
 - command line option, [134](#)
- session-keyalg
 - command line option, [134](#)
- session-keyfile
 - command line option, [134](#)
- session-keyname
 - command line option, [134](#)
- severity
 - command line option, [115](#)
- show
 - command line option, [90](#)
- sig-signing-nodes
 - command line option, [162](#)
- sig-signing-signatures
 - command line option, [162](#)
- sig-signing-type
 - command line option, [162](#)
- sig-validity-interval
 - command line option, [163](#)
- sign
 - command line option, [28](#)
- signing
 - command line option, [28](#)
- stacksize
 - command line option, [135](#)
- statistics-file
 - command line option, [135](#)

stats
 command line option, [29](#)
status
 command line option, [29](#)
stderr
 command line option, [115](#)
stop
 command line option, [29](#)
sync
 command line option, [29](#)
syslog
 command line option, [114](#)
T
tcp-clients
 command line option, [135](#)
tcp-listen-queue
 command line option, [135](#)
tcp-only
 command line option, [168](#)
thaw
 command line option, [29](#)
tkey-dhkey
 command line option, [135](#)
tkey-domain
 command line option, [136](#)
tkey-gssapi-credential
 command line option, [136](#)
tkey-gssapi-keytab
 command line option, [136](#)
trace
 command line option, [29](#)
transfer-source
 command line option, [163](#), [168](#)
transfer-source-v6
 command line option, [163](#), [168](#)
transfers
 command line option, [168](#)
transfers-in
 command line option, [136](#)
transfers-out
 command line option, [136](#)
transfers-per-ns
 command line option, [137](#)
try-tcp-refresh
 command line option, [164](#)
tsig-delete
 command line option, [30](#)

tsig-list
 command line option, [30](#)
ttl
 command line option, [89](#)

U

update
 command line option, [90](#)
update-check-ksk
 command line option, [164](#)
use-alt-transfer-source
 command line option, [164](#)
use-v4-udp-ports
 command line option, [137](#)
use-v6-udp-ports
 command line option, [137](#)

V

validation
 command line option, [30](#)
version
 command line option, [91](#), [137](#)

Z

zero-no-soa-ttl
 command line option, [164](#)
zero-no-soa-ttl-cache
 command line option, [152](#)
zone
 command line option, [89](#)
zone-statistics
 command line option, [164](#)
zonestatus
 command line option, [30](#)