
ISC


BIND 9

BIND 9 Administrator Reference Manual

Internet Systems Consortium

Aug 20, 2020

Contents

CHAPTER 1

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

The Berkeley Internet Name Domain (BIND) implements a domain name server for a number of operating systems. This document provides basic information about the installation and care of the Internet Systems Consortium (ISC) BIND version 9 software package for system administrators.

This manual covers BIND version .

1.2 Organization of This Document

In this document, *Chapter 1* introduces the basic DNS and BIND concepts. *Chapter 2* describes resource requirements for running BIND 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 BIND 9 software. The task-oriented section is followed by *Chapter 4*, which is organized as a reference manual to aid in the ongoing maintenance of the software. *Chapter 5* contains more advanced concepts that the system administrator may need for implementing certain options. *Chapter 6* addresses security considerations, and *Chapter 7* 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 BIND and the Domain Name System.

1.3 Conventions Used in This Document

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

To describe:	*We use the style:*
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 BIND configuration file:

To describe:	*We use the style:*
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 BIND (Berkeley Internet Name Domain) software package, and we begin by reviewing the fundamentals of the Domain Name System (DNS) as they relate to BIND.

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 BIND 9 software distribution contains a name server, `named`, and a set of associated tools.

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 *Types of Resource Records and When to Use Them*.

For more detailed information about the design of the DNS and the DNS protocol, please refer to the standards documents listed in *Request for Comments (RFCs)*.

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 **RFC 1033**, **RFC 1034** and **RFC 1035** to gain a complete understanding of this difficult and subtle topic.

Though BIND 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` (*Diagnostic Tools*).

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.

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.

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.

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*.

Forwarders are typically used when an administrator does not wish for all the servers at a given site to interact directly with the rest of the Internet. For example, a common scenario is when multiple internal DNS servers are behind an Internet firewall. Servers behind the firewall forward their requests to the server with external access, which queries Internet DNS servers on the internal servers' behalf.

Another scenario (largely now superseded by Response Policy Zones) is to send queries first to a custom server for RBL processing before forwarding them to the wider Internet.

There may be one or more forwarders in a given setup. The order in which the forwarders are listed in `named.conf` does not determine the sequence in which they are queried; rather, `named` uses the response times from previous queries to select the server that is likely to respond the most quickly. A server that has not yet been queried is given an initial small random response time to ensure that it is tried at least once. Dynamic adjustment of the recorded response times ensures that all forwarders are queried, even those with slower response times. This permits changes in behavior based on server responsiveness.

1.4.6 Name Servers in Multiple Roles

The BIND 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.

CHAPTER 2

BIND Resource Requirements

2.1 Hardware Requirements

DNS hardware requirements have traditionally been quite modest. For many installations, servers that have been retired from active duty have performed admirably as DNS servers.

However, the DNSSEC features of BIND 9 may prove to be quite CPU intensive, so organizations that make heavy use of these features may wish to consider larger systems for these applications. BIND 9 is fully multithreaded, allowing full utilization of multiprocessor systems for installations that need it.

2.2 CPU Requirements

CPU requirements for BIND 9 range from i386-class machines, for serving static zones without caching, to enterprise-class machines to process many dynamic updates and DNSSEC-signed zones, serving many thousands of queries per second.

2.3 Memory Requirements

Server memory must be sufficient to hold both the cache and the zones loaded from disk. The `max-cache-size` option 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; unfortunately, the best way to determine this for a given installation is to watch the name server in operation. After a few weeks the server process should reach a relatively stable size where entries are expiring from the cache as fast as they are being inserted.

2.4 Name Server Intensive Environment Issues

For name server intensive environments, there are two alternative configurations that may be used. The first is one where clients and any second-level internal name servers query a main name server, which has enough memory to build a large cache; this approach minimizes the bandwidth used by external name lookups. The second alternative is to set up second-level internal name servers to make queries independently. In this configuration, none of the individual machines needs to have as much memory or CPU power as in the first alternative, but this has the disadvantage of making many more external queries, as none of the name servers share their cached data.

2.5 Supported Operating Systems

ISC BIND 9 compiles and runs on a large number of Unix-like operating systems and on Microsoft Windows Server 2012 R2, 2016, and Windows 10. For an up-to-date list of supported systems, see the PLATFORMS.md file in the top-level directory of the BIND 9 source distribution.

CHAPTER 3

Name Server Configuration

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 primary (master) server for `example.com` and a secondary (slave) server for the subdomain `eng.example.com`.

```
options {
     // Working directory
     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, assuming three HTTP servers 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	CLASS	TYPE	Resource Record (RR) Data
www	600	IN	A	10.0.0.1
	600	IN	A	10.0.0.2
	600	IN	A	10.0.0.3

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

For more detail on ordering responses, check the `rrset-order` sub-statement in the `options` statement; 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.

Diagnostic Tools

The `dig`, `host`, and `nslookup` programs are all 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.

`nslookup` `nslookup` has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

```
nslookup [-option][ [host-to-find]|[-[server]] ]
```

Interactive mode is entered when no arguments are given (the default name server is used) or when the first argument is a hyphen (`-`) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Due to its arcane user interface and frequently inconsistent behavior, we do not recommend the use of `nslookup`. Use `dig` instead.

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 [-djqvD][-c class][-o output][-t directory][-w directory][-k
(ignore|warn|fail)][-n (ignore|warn|fail)][-W (ignore|warn)] zone [filename]
```

`named-compilezone` This tool is similar to `named-checkzone,` but it always dumps the zone content to a specified file (typically in a different format).

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

```
rndc [-c config][-s server][-p port][-y key] command [command...]
```

See *rndc - name server control utility* 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/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/rndc.key` (or whatever `sysconfdir` was defined when the BIND build was configured). The `rndc.key` file is generated by running `rndc-confgen -a` as described in *controls Statement Definition and Usage*.

The format of the configuration file is similar to that of `named.conf`, but is 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 the 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/rndc.conf`, allows 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 is running with the following controls statements:

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

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

Running the `rndc-confgen` program conveniently creates a `rndc.conf` file, and also displays the corresponding `controls` statement needed to add to `named.conf`. Alternatively, it is possible to run `rndc-confgen -a` to set up a `rndc.key` file and not modify `named.conf` at all.

3.3.2 Signals

Certain UNIX signals cause the name server to take specific actions, as described in the following table. These signals can be sent using the `kill` command.

`SIGHUP`	Causes the server to read `named.conf` and reload the database.
`SIGTERM`	Causes the server to clean up and exit.
`SIGINT`	Causes the server to clean up and exit.

3.4 Plugins

Plugins are a mechanism to extend the functionality of `named` using dynamically loadable libraries. By using plugins, core server functionality can be kept simple for the majority of users; more complex code implementing optional features need only be installed by users that need those features.

The plugin interface is a work in progress, and is expected to evolve as more plugins are added. Currently, only "query plugins" are supported; these modify the name server query logic. Other plugin types may be added in the future.

The only plugin currently included in BIND is `filter-aaaa.so`, which replaces the `filter-aaaa` feature that previously existed natively as part of `named`. The code for this feature has been removed from `named` and can no longer be configured using standard `named.conf` syntax, but linking in the `filter-aaaa.so` plugin provides identical functionality.

3.4.1 Configuring Plugins

A plugin is configured with the `plugin` statement in `named.conf`:

```
plugin query "library.so" {
    parameters
};
```

In this example, file `library.so` is the plugin library. `query` indicates that this is a query plugin.

Multiple `plugin` statements can be specified, to load different plugins or multiple instances of the same plugin.

`parameters` are passed as an opaque string to the plugin's initialization routine. Configuration syntax will differ depending on the module.

3.4.2 Developing Plugins

Each plugin implements four functions:

- plugin_register to allocate memory, configure a plugin instance, and attach to hook points within named ,
- plugin_destroy to tear down the plugin instance and free memory,
- plugin_version to check that the plugin is compatible with the current version of the plugin API,
- plugin_check to test syntactic correctness of the plugin parameters.

At various locations within the `named` source code, there are "hook points" at which a plugin may register itself. When a hook point is reached while `named` is running, it is checked to see whether any plugins have registered themselves there; if so, the associated "hook action" is called - this is a function within the plugin library. Hook actions may examine the runtime state and make changes - for example, modifying the answers to be sent back to a client or forcing a query to be aborted. More details can be found in the file `lib/ns/include/ns/hooks.h`.

CHAPTER 4

BIND 9 Configuration Reference

4.1 Configuration File Elements

Following is a list of elements used throughout the BIND configuration file documentation:

`acl_name` The name of an `address_match_list` as defined by the `acl` statement.

`address_match_list` A list of one or more `ip_addr`, `ip_prefix`, `key_id`, or `acl_name` elements, see *Address Match Lists*.

`masters_list` A named list of one or more `ip_addr` with optional `key_id` and/or `ip_port`. A `masters_list` may include other `masters_lists`.

`domain_name` A quoted string which is used as a DNS name, for example "`my.test.domain`".

`namelist` A list of one or more `domain_name` elements.

`dotted_decimal` One to four integers valued 0 through 255 separated by dots ('.'), such as `123`, `45.67` or `89.123.45.67`.

`ip4_addr` An IPv4 address with exactly four elements in `dotted_decimal` notation.

`ip6_addr` An IPv6 address, such as `2001:db8::1234`. IPv6 scoped addresses that have ambiguity on their scope zones must be disambiguated by an appropriate zone ID with the percent character ('%') as delimiter. It is strongly recommended to use string zone names rather than numeric identifiers, to be robust against system configuration changes. However, since there is no standard mapping for such names and identifier values, currently only interface names as link identifiers are supported, assuming one-to-one mapping between interfaces and links. For example, a link-local address `fe80::1` on the link attached to the interface `ne0` can be specified as `fe80::1%ne0`. Note that on most systems link-local addresses always have ambiguity and need to be disambiguated.

`ip_addr` An `ip4_addr` or `ip6_addr`.

`ip_dscp` A `number` between 0 and 63, used to select a differentiated services code point (DSCP) value for use with outgoing traffic on operating systems that support DSCP.

ip_port An IP port `number`. The `number` is limited to 0 through 65535, with values below 1024 typically restricted to use by processes running as root. In some cases, an asterisk (`*`) character can be used as a placeholder to select a random high-numbered port.

ip_prefix An IP network specified as an `ip_addr`, followed by a slash ('/') and then the number of bits in the netmask. Trailing zeros in an"ip_addr" may omitted. For example, `127/8` is the network `127.0.0.0` with network `1.2.3.0` with netmask `255.255.255.240`. When specifying a prefix involving a IPv6 scoped address the scope may be omitted. In that case the prefix matches packets from any scope.

key_id A `domain_name` representing the name of a shared key, to be used for transaction security.

key_list A list of one or more `key_id`s, separated by semicolons and ending with a semicolon.

number A non-negative 32-bit integer (i.e., a number between 0 and 4294967295, inclusive). Its acceptable value might be further limited by the context in which it is used.

fixedpoint A non-negative real number that can be specified to the nearest one-hundredth. Up to five digits can be specified before a decimal point, and up to two digits after, so the maximum value is 99999.99. Acceptable values might be further limited by the contexts in which they are used.

path_name A quoted string which is used as a pathname, such as `zones/master/my.test.domain`.

port_list A list of an `ip_port` or a port range. A port range is specified in the form of `range` followed by two `ip_port`s, `port_low` and `port_high`, which represents port numbers from `port_low` through `port_high`, inclusive. `port_low` must not be larger than `port_high`. For example, `range 1024 65535` represents ports from 1024 through 65535. In either case an asterisk ('*') character is not allowed as a valid `ip_port`.

size_spec A 64-bit unsigned integer, or the keywords `unlimited` or `default`. Integers may take values 0 <= value <= 18446744073709551615, though certain parameters (such as `max-journal-size`) may use a more limited range within these extremes. In most cases, setting a value to 0 does not literally mean zero; it means "undefined" or "as big as possible", depending on the context. See the explanations of particular parameters that use `size_spec` for details on how they interpret its use. Numeric values can optionally be followed by a scaling factor: `K` or `k` for kilobytes, `M` or `m` for megabytes, and `G` or `g` for gigabytes, which scale by 1024, 1024*1024, and 1024*1024*1024 respectively. `unlimited` generally means "as big as possible," and is usually the best way to safely set a very large number. `default` uses the limit that was in force when the server was started.

size_or_percent A `size_spec` or integer value followed by '%' to represent percent. The behavior is exactly the same as `size_spec`, but `size_or_percent` also allows specifying a positive integer value followed by the '%' sign to represent percent.

yes_or_no Either `yes` or `no`. The words `true` and `false` are also accepted, as are the numbers `1` and `0`.

dialup_option One of `yes`, `no`, `notify`, `notify-passive`, `refresh`, or `passive`. When used in a zone, `notify-passive`, `refresh`, and `passive` are restricted to secondary and stub zones.

4.1.1 Address Match Lists

Syntax

```
address_match_list = address_match_list_element ; ...

address_match_list_element = [ ! ] ( ip_address | ip_prefix |
     key key_id | acl_name | { address_match_list } )
```

Definition and Usage

Address match lists are primarily used to determine access control for various server operations. They are also used in the `listen-on` and `sortlist` statements. The elements which constitute an address match 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 list defined with the `acl` statement
- a nested address match list enclosed in braces

Elements can be negated with a leading exclamation mark (`!`), and the match list names "any", "none", "localhost", and "localnets" are predefined. More information on those names can be found in the description of the `acl` statement.

The addition of the key clause made the name of this syntactic element something of a misnomer, since security keys can be used to validate access without regard to a host or network address. Nonetheless, the term "address match list" is still used throughout the documentation.

When a given IP address or prefix is compared to an address match list, the comparison takes place in approximately O(1) time. However, key comparisons require that the list of keys be traversed until a matching key is found, and therefore may be somewhat slower.

The interpretation of a match depends on whether the list is being used for access control, defining `listen-on` ports, or in a `sortlist`, and whether the 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`, `blackhole`, and `keep-response-order` all use address match lists. Similarly, the `listen-on` option causes the server to refuse queries on any of the machine's addresses which do not match the list.

Order of insertion is significant. If more than one element in an ACL is found to match a given IP address or prefix, preference is 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 matches 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 pass through.

4.1.2 Comment Syntax

The BIND 9 comment syntax allows for comments to appear anywhere that whitespace may appear in a BIND configuration file. To appeal to programmers of all kinds, they can be written in the C, C++, or shell/perl style.

Syntax

```
/* This is a BIND comment as in C */
```

```
// This is a BIND comment as in C++
```

```
# This is a BIND comment as in common Unix shells
# and perl
```

Definition and Usage

Comments may appear anywhere that whitespace may appear in a BIND configuration file.

C-style comments start with the two characters /* (slash, star) and end with */ (star, slash). Because they are completely delimited with these characters, they can be used to comment only a portion of a line or to span multiple lines.

C-style comments cannot be nested. For example, the following 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 the start of a comment.  The next line
// is a new comment, even though it is logically
// part of the previous comment.
```

Shell-style (or perl-style, if you prefer) comments start with the character `#` (number sign) and continue to the end of the physical line, as in C++ comments. For example:

```
# This is the start of a comment.  The next line
# is a new comment, even though it is logically
# part of the previous comment.
```

Warning: The semicolon (`;`) character cannot start a comment, unlike in a zone file. The semicolon indicates the end of a configuration statement.

4.2 Configuration File Grammar

A BIND 9 configuration consists of 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.

The following statements are supported:

`acl` Defines a named IP address matching list, for access control and other uses.

`controls` Declares control channels to be used by the `rndc` utility.

dnssec-policy Describes a DNSSEC key and signing policy for zones. See *dnssec-policy Grammar* for details.

include Includes a file.

key Specifies key information for use in authentication and authorization using TSIG.

logging Specifies what the server logs, and where the log messages are sent.

masters Defines a named masters list for inclusion in stub and secondary zones' `masters` or `also-notify` lists.

options Controls global server configuration options and sets defaults for other statements.

server Sets certain configuration options on a per-server basis.

statistics-channels Declares communication channels to get access to `named` statistics.

trust-anchors Defines DNSSEC trust anchors: if used with the `initial-key` or `initial-ds` keyword, trust anchors are kept up-to-date using **RFC 5011** trust anchor maintenance; if used with `static-key` or `static-ds`, keys are permanent.

managed-keys Is identical to `trust-anchors`; this option is deprecated in favor of `trust-anchors` with the `initial-key` keyword, and may be removed in a future release.

trusted-keys Defines permanent trusted DNSSEC keys; this option is deprecated in favor of `trust-anchors` with the `static-key` keyword, and may be removed in a future release.

view Defines a view.

zone Defines a zone.

The `logging` and `options` statements may only occur once per configuration.

4.2.1 `acl` Statement Grammar

```
acl <string> { <address_match_element>; ... };
```

4.2.2 `acl` Statement Definition and Usage

The `acl` statement assigns a symbolic name to an address match list. It gets its name from a primary use of address match lists: Access Control Lists (ACLs).

The following ACLs are built-in:

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`.

4.2.3 `controls` Statement Grammar

```
controls {
      inet ( <ipv4_address> | <ipv6_address> |
          * ) [ port ( <integer> | * ) ] allow
          { <address_match_element>; ... } [
          keys { <string>; ... } ] [ read-only
          <boolean> ];
      unix <quoted_string> perm <integer>
          owner <integer> group <integer> [
          keys { <string>; ... } ] [ read-only
          <boolean> ];
};
```

4.2.4 `controls` Statement Definition and Usage

The `controls` statement declares control channels to be used by system administrators to control the operation of the name server. These control channels are used by the `rndc` utility to send commands to and retrieve non-DNS results from a name server.

An `inet` control channel is a TCP socket listening at the specified `ip_port` on the specified `ip_addr`, which can be an IPv4 or IPv6 address. An `ip_addr` of `*` (asterisk) is interpreted as the IPv4 wildcard address; connections are accepted on any of the system's IPv4 addresses. To listen on the IPv6 wildcard address, use an `ip_addr` of `::`. If `rndc` is only used on the local host, using the loopback address (`127.0.0.1` or `::1`) is recommended for maximum security.

If no port is specified, port 953 is used. The asterisk "`*`" cannot be used for `ip_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_list`. This is for simple IP address-based filtering only; any `key_id` elements of the `address_match_list` are ignored.

A `unix` control channel is a Unix domain socket listening at the specified path in the file system. Access to the socket is specified by the `perm`, `owner`, and `group` clauses. Note that on some platforms (SunOS and Solaris), the permissions (`perm`) are applied to the parent directory as the permissions on the socket itself are ignored.

The primary authorization mechanism of the command channel is the `key_list`, which contains a list of `key_id`s. Each `key_id` in the `key_list` is authorized to execute commands over the control channel. See *Administrative Tools* for information about configuring keys in `rndc`.

If the `read-only` clause is enabled, the control channel is limited to the following set of read-only commands: `nta -dump`, `null`, `status`, `showzone`, `testgen`, and `zonestatus`. By default, `read-only` is not enabled and the control channel allows read-write access.

If no `controls` statement is present, `named` sets 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` attempts to load the command channel key from the file `rndc.key` in `/etc` (or whatever `sysconfdir` was specified as when BIND was built). To create a `rndc.key` file, run `rndc-confgen -a`.

To disable the command channel, use an empty `controls` statement: `controls { };`.

4.2.5 `include` Statement Grammar

```
include filename;
```

4.2.6 `include` Statement Definition and Usage

The `include` statement inserts the specified file (or files if a valid glob expression is detected) at the point where the `include` statement is encountered. The `include` statement facilitates the administration of configuration files by permitting the reading or writing of some things but not others. For example, the statement could include private keys that are readable only by the name server.

4.2.7 `key` Statement Grammar

```
key <string> {
      algorithm <string>;
      secret <string>;
};
```

4.2.8 `key` Statement Definition and Usage

The `key` statement defines a shared secret key for use with TSIG (see *TSIG*) or the command channel (see *controls Statement Definition and Usage*).

The `key` statement can occur at the top level of the configuration file or inside a `view` statement. Keys defined in top-level `key` statements can be used in all views. Keys intended for use in a `controls` statement (see *controls Statement Definition and Usage*) must be defined at the top level.

The key_id, also known as the key name, is a domain name uniquely identifying the key. It can be used in a `server` statement to cause requests sent to that server to be 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_id is a string that specifies a security/authentication algorithm. The `named` server supports `hmac-md5`, `hmac-sha1`, `hmac-sha224`, `hmac-sha256`, `hmac-sha384`, and `hmac-sha512` TSIG authentication. Truncated hashes are supported by appending the minimum number of required bits preceded by a dash, e.g., `hmac-sha1-80`. The secret_string is the secret to be used by the algorithm, and is treated as a Base64 encoded string.

4.2.9 `logging` Statement Grammar

```
logging {
      category <string> { <string>; ... };
      channel <string> {
              buffered <boolean>;
              file <quoted_string> [ versions ( unlimited | <integer> ) ]
                  [ size <size> ] [ suffix ( increment | timestamp ) ];
              null;
              print-category <boolean>;
              print-severity <boolean>;
              print-time ( iso8601 | iso8601-utc | local | <boolean> );
```

(continues on next page)

(continued from previous page)

```
            severity <log_severity>;
            stderr;
            syslog [ <syslog_facility> ];
      };
};
```

4.2.10 `logging` Statement Definition and Usage

The `logging` statement configures a wide variety of logging options for the name server. 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 used to define as many channels and categories as desired. If there is no `logging` statement, the logging configuration is:

```
logging {
     category default { default_syslog; default_debug; };
     category unmatched { null; };
};
```

If `named` is started with the `-L` option, it logs to the specified file at startup, instead of using syslog. In this case the logging configuration is:

```
logging {
     category default { default_logfile; default_debug; };
     category unmatched { null; };
};
```

The logging configuration is only established when the entire configuration file has been parsed. When the server starts up, all logging messages regarding syntax errors in the configuration file go to the default channels, or to standard error if the `-g` option was specified.

The `channel` Phrase

All log output goes to one or more `channels`; there is no limit to the number of channels that can be created.

Every channel definition must include a destination clause that says whether messages selected for the channel go to a file, go to a particular syslog facility, go to the standard error stream, or are discarded. The definition can optionally also limit the message severity level that is 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).

The `null` destination clause causes all messages sent to the channel to be discarded; in that case, other options for the channel are meaningless.

The `file` destination clause directs the channel to a disk file. It can include additional arguments to specify how large the file is allowed to become before it is rolled to a backup file (`size`), how many backup versions of the file are saved each time this happens (`versions`), and the format to use for naming backup versions (`suffix`).

The `size` option is used to limit log file growth. If the file ever exceeds the specified size, then `named` stops writing to the file unless it has a `versions` option associated with it. If backup versions are kept, the files are rolled as described below. If there is no `versions` option, no more data is 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.

File rolling only occurs when the file exceeds the size specified with the `size` option. No backup versions are kept by default; any existing log file is simply appended. The `versions` option specifies how many backup versions of the file should be kept. If set to `unlimited`, there is no limit.

The `suffix` option can be set to either `increment` or `timestamp`. If set to `timestamp`, then when a log file is rolled, it is saved with the current timestamp as a file suffix. If set to `increment`, then backup files are saved with incrementing numbers as suffixes; older files are renamed when rolling. For example, if `versions` is set to 3 and `suffix` to `increment`, then when `filename.log` reaches the size specified by `size`, `filename.log.1` is renamed to `filename.log.2`, `filename.log.0` is renamed to `filename.log.1`, and `filename.log` is renamed to `filename.log.0`, whereupon a new `filename.log` is opened.

Example usage of the `size`, `versions`, and `suffix` options:

```
channel an_example_channel {
    file "example.log" versions 3 size 20m suffix increment;
    print-time yes;
    print-category yes;
};
```

The `syslog` destination clause directs the channel to the system log. Its argument is a syslog facility as described in the `syslog` man page. 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 `syslog` handles messages sent to this facility is described in the `syslog.conf` man page. On a system which uses a very old version of `syslog`, that only uses two arguments to the `openlog()` function, this clause is silently ignored.

On Windows machines syslog messages are directed to the EventViewer.

The `severity` clause works like `syslog`'s "priorities", except that they can also be used when writing straight to a file rather than using `syslog`. Messages which are not at least of the severity level given are not selected for the channel; messages of higher severity levels are accepted.

When using `syslog`, the `syslog.conf` priorities also determine what eventually passes through. For example, defining a channel facility and severity as `daemon` and `debug`, but only logging `daemon.warning` via `syslog.conf`, causes messages of severity `info` and `notice` to be dropped. If the situation were reversed, with `named` writing messages of only `warning` or higher, then `syslogd` would print all messages it received from the channel.

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 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, debugging mode is active. The global debug level is set either by starting the `named` server with the `-d` flag followed by a positive integer, or by running `rndc trace`. 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; 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;
};
```

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.

`print-time` can be set to `yes`, `no`, or a time format specifier, which may be one of `local`, `iso8601`, or `iso8601-utc`. If set to `no`, the date and time are not logged. If set to `yes` or `local`, the date and time are logged in a human-readable format, using the local time zone. If set to `iso8601` the local time is logged in ISO 8601 format. If set to `iso8601-utc`, the date and time are logged in ISO 8601 format, with time zone set to UTC. The default is `no`.

`print-time` may be specified for a `syslog` channel, but it is usually pointless since `syslog` also logs the date and time.

If `print-category` is requested, then the category of the message is logged as well. Finally, if `print-severity` is on, then the severity level of the message is logged. The `print-` options may be used in any combination, and are always printed in the following order: time, category, severity. Here is an example where all three `print-` options are on:

```
28-Feb-2000 15:05:32.863 general: notice: running
```

If `buffered` has been turned on, the output to files is not flushed after each log entry. By default all log messages are flushed.

There are four predefined channels that are used for `named`'s default logging, as follows. If `named` is started with `-L` then a fifth channel `default_logfile` is added. How they are used is described in *The category Phrase*.

```
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 '-g' 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;
};

channel default_logfile {
    // this channel is only present if named is
    // started with the -L option, whose argument
```

(continues on next page)

(continued from previous page)

```
    // provides the file name
    file "...";
    // log at the server's current debug level
    severity dynamic;
};
```

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 `-u` command line option 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. To capture this output, run the server with the `-L` option to specify a default logfile, or the `-g` option to log to standard error which can be redirected to a file.

Once a channel is defined, it cannot be redefined. The built-in channels cannot be altered directly, but the default logging can be modified by pointing categories at defined channels.

The `category` Phrase

There are many categories, so desired logs can be sent anywhere while unwanted logs are ignored. If a list of channels is not specified for a category, log messages in that category are sent to the `default` category instead. If no default category is specified, the following "default default" is used:

```
category default { default_syslog; default_debug; };
```

If `named` is started with the `-L` option then the default category is:

```
category default { default_logfile; default_debug; };
```

As an example, let's say a user wants to log security events to a file, but also wants to keep the default logging behavior. They would 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; };
```

Following are the available categories and brief descriptions of the types of log information they contain. More categories may be added in future BIND releases.

`client` Processing of client requests.

`cname` 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` Logging options for those categories where no specific configuration has been defined.

`delegation-only` 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.

`dnstap` The "dnstap" DNS traffic capture system.

`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 non-**RFC 1034** 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 DNSSEC validation, which requires EDNS for the DNSSEC records to be returned.

`general` Catch-all for many things that still are not classified into categories.

`lame-servers` Misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.

`network` Network operations.

`notify` The NOTIFY protocol.

`nsid` NSID options received from upstream servers.

`queries` Location where queries should be logged.

At startup, specifying the category `queries` also enables query logging unless the `querylog` option has been specified.

The query log entry first reports a client object identifier in @0x<hexadecimal-number> format. Next, it reports the client's IP address and port number, and the query name, class, and type. Next, it reports whether the Recursion Desired flag was set (+ if set, - if not set), whether the query was signed (S), whether EDNS was in use along with the EDNS version number (E(#)), whether TCP was used (T), whether DO (DNSSEC Ok) was set (D), whether CD (Checking Disabled) was set (C), whether a valid DNS Server COOKIE was received (V), and whether a DNS COOKIE option without a valid Server COOKIE was present (K). After this, the destination address the query was sent to is reported. Finally, if any CLIENT-SUBNET option was present in the client query, it is included in square brackets in the format [ECS address/source/scope].

```
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.

`rate-limit` Start, periodic, and final notices of the rate limiting of a stream of responses that 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 limiting stops. A lack of memory can hurry the final notice, which is indicated by an initial asterisk (*). Various internal events are logged at debug 1 level 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.

`serve-stale` Indication of whether a stale answer is used following a resolver failure.

`spill` Queries that have been terminated, either by dropping or responding with SERVFAIL, as a result of a fetchlimit quota being exceeded.

`trust-anchor-telemetry` Trust-anchor-telemetry requests received by `named`.

`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` Dynamic updates.

`update-security` Approval and denial of update requests.

`xfer-in` Zone transfers the server is receiving.

`xfer-out` Zone transfers the server is sending.

`zoneload` Loading of zones and creation of automatic empty zones.

The `query-errors` Category

The `query-errors` category is used to indicate why and how specific queries resulted in responses which indicate an error. Normally, these messages are logged at `debug` logging levels; note, however, that if query logging is active, some are logged at `info`. The logging levels are described below:

At `debug` levels of 1 or higher - or at `info` when query logging is active - 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 file `query.c`. Log messages of this level are particularly helpful in identifying the cause of SERVFAIL for an authoritative server.

At `debug` level 2 or higher, detailed context information about recursive resolutions that resulted in SERVFAIL is logged. The log message looks like this:

```
fetch completed at resolver.c:2970 for www.example.com/A
in 10.000183: timed out/success [domain:example.com,
referral:2,restart:7,qrysent:8,timeout:5,lame:0,quota: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 10.000183 seconds, and the final result that led to the SERVFAIL was determined at line 2970 of source file `resolver.c`.

The next part shows the detected final result and the latest result of DNSSEC validation. The latter is always "success" when no validation attempt was made. In this example, this query probably resulted in SERVFAIL because all name servers are down or unreachable, leading to a timeout in 10 seconds. DNSSEC validation was probably not attempted.

The last part, enclosed in square brackets, shows statistics collected for this particular resolution attempt. 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 list.

`referral` The number of referrals the resolver received throughout the resolution process. In the above `example.com` there are two.

`restart` The number of cycles that the resolver tried remote servers at the `domain` zone. In each cycle the resolver sends one query (possibly resending it, depending on the response) to each known name server of the `domain` zone.

`qrysent` The number of queries the resolver sent at the `domain` zone.

`timeout` The number of timeouts the resolver received since the last response.

`lame` The number of lame servers the resolver detected at the `domain` zone. A server is detected to be lame either by an invalid response or as a result of lookup in BIND 9's address database (ADB), where lame servers are cached.

`quota` The number of times the resolver was unable to send a query because it had exceeded the permissible fetch quota for a server.

`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 server addresses of the"domain" zone in the ADB. One common case of this is that the remote server's name does not have any address records.

`findfail` Failures of resolving remote server addresses. This is a total number of failures throughout the eesolution 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 `debug` level 3 or higher, the same messages as those at `debug` level 1 are logged for errors other than SERVFAIL. Note that negative responses such as NXDOMAIN are not errors, and are not logged at this debug level.

At `debug` level 4 or higher, the detailed context information logged at `debug` level 2 is logged for errors other than SERVFAIL and for negative responses such as NXDOMAIN.

4.2.11 `masters` Statement Grammar

```
masters <string> [ port <integer> ] [ dscp
    <integer> ] { ( <masters> | <ipv4_address> [
    port <integer> ] | <ipv6_address> [ port
    <integer> ] ) [ key <string> ]; ... };
```

4.2.12 `masters` Statement Definition and Usage

`masters` lists allow for a common set of masters to be easily used by multiple stub and secondary zones in their `masters` or `also-notify` lists.

4.2.13 `options` Statement Grammar

This is the grammar of the `options` statement in the `named.conf` file:

```
options {
      allow-new-zones <boolean>;
      allow-notify { <address_match_element>; ... };
      allow-query { <address_match_element>; ... };
      allow-query-cache { <address_match_element>; ... };
      allow-query-cache-on { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      allow-recursion { <address_match_element>; ... };
      allow-recursion-on { <address_match_element>; ... };
      allow-transfer { <address_match_element>; ... };
      allow-update { <address_match_element>; ... };
      allow-update-forwarding { <address_match_element>; ... };
      also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> |
          <ipv4_address> [ port <integer> ] | <ipv6_address> [ port
          <integer> ] ) [ key <string> ]; ... };
      alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * )
          ] [ dscp <integer> ];
      alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> |
          * ) ] [ dscp <integer> ];
      answer-cookie <boolean>;
      attach-cache <string>;
      auth-nxdomain <boolean>; // default changed
      auto-dnssec ( allow | maintain | off );
      automatic-interface-scan <boolean>;
      avoid-v4-udp-ports { <portrange>; ... };
      avoid-v6-udp-ports { <portrange>; ... };
      bindkeys-file <quoted_string>;
      blackhole { <address_match_element>; ... };
      cache-file <quoted_string>;
      catalog-zones { zone <string> [ default-masters [ port <integer> ]
          [ dscp <integer> ] { ( <masters> | <ipv4_address> [ port
          <integer> ] | <ipv6_address> [ port <integer> ] ) [ key
          <string> ]; ... } ] [ zone-directory <quoted_string> ] [
          in-memory <boolean> ] [ min-update-interval <duration> ]; ... };
      check-dup-records ( fail | warn | ignore );
      check-integrity <boolean>;
      check-mx ( fail | warn | ignore );
      check-mx-cname ( fail | warn | ignore );
      check-names ( primary | master |
          secondary | slave | response ) (
          fail | warn | ignore );
      check-sibling <boolean>;
      check-spf ( warn | ignore );
```

(continues on next page)

(continued from previous page)

```
check-srv-cname ( fail | warn | ignore );
check-wildcard <boolean>;
clients-per-query <integer>;
cookie-algorithm ( aes | siphash24 );
cookie-secret <string>;
coresize ( default | unlimited | <sizeval> );
datasize ( default | unlimited | <sizeval> );
deny-answer-addresses { <address_match_element>; ... } [
    except-from { <string>; ... } ];
deny-answer-aliases { <string>; ... } [ except-from { <string>; ...
    } ];
dialup ( notify | notify-passive | passive | refresh | <boolean> );
directory <quoted_string>;
disable-algorithms <string> { <string>;
    ... };
disable-ds-digests <string> { <string>;
    ... };
disable-empty-zone <string>;
dns64 <netprefix> {
        break-dnssec <boolean>;
        clients { <address_match_element>; ... };
        exclude { <address_match_element>; ... };
        mapped { <address_match_element>; ... };
        recursive-only <boolean>;
        suffix <ipv6_address>;
};
dns64-contact <string>;
dns64-server <string>;
dnskey-sig-validity <integer>;
dnsrps-enable <boolean>;
dnsrps-options { <unspecified-text> };
dnssec-accept-expired <boolean>;
dnssec-dnskey-kskonly <boolean>;
dnssec-loadkeys-interval <integer>;
dnssec-must-be-secure <string> <boolean>;
dnssec-policy <string>;
dnssec-secure-to-insecure <boolean>;
dnssec-update-mode ( maintain | no-resign );
dnssec-validation ( yes | no | auto );
dnstap { ( all | auth | client | forwarder |
    resolver | update ) [ ( query | response ) ];
    ... };
dnstap-identity ( <quoted_string> | none |
    hostname );
dnstap-output ( file | unix ) <quoted_string> [
    size ( unlimited | <size> ) ] [ versions (
    unlimited | <integer> ) ] [ suffix ( increment
    | timestamp ) ];
dnstap-version ( <quoted_string> | none );
dscp <integer>;
dual-stack-servers [ port <integer> ] { ( <quoted_string> [ port
    <integer> ] [ dscp <integer> ] | <ipv4_address> [ port
```

(continues on next page)

(continued from previous page)

```
        <integer> ] [ dscp <integer> ] | <ipv6_address> [ port
        <integer> ] [ dscp <integer> ] ); ... };
    dump-file <quoted_string>;
    edns-udp-size <integer>;
    empty-contact <string>;
    empty-server <string>;
    empty-zones-enable <boolean>;
    fetch-quota-params <integer> <fixedpoint> <fixedpoint> <fixedpoint>;
    fetches-per-server <integer> [ ( drop | fail ) ];
    fetches-per-zone <integer> [ ( drop | fail ) ];
    files ( default | unlimited | <sizeval> );
    flush-zones-on-shutdown <boolean>;
    forward ( first | only );
    forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address>
        | <ipv6_address> ) [ port <integer> ] [ dscp <integer> ]; ... };
    fstrm-set-buffer-hint <integer>;
    fstrm-set-flush-timeout <integer>;
    fstrm-set-input-queue-size <integer>;
    fstrm-set-output-notify-threshold <integer>;
    fstrm-set-output-queue-model ( mpsc | spsc );
    fstrm-set-output-queue-size <integer>;
    fstrm-set-reopen-interval <duration>;
    geoip-directory ( <quoted_string> | none );
    glue-cache <boolean>;
    heartbeat-interval <integer>;
    hostname ( <quoted_string> | none );
    inline-signing <boolean>;
    interface-interval <duration>;
    ixfr-from-differences ( primary | master | secondary | slave |
        <boolean> );
    keep-response-order { <address_match_element>; ... };
    key-directory <quoted_string>;
    lame-ttl <duration>;
    listen-on [ port <integer> ] [ dscp
        <integer> ] {
        <address_match_element>; ... };
    listen-on-v6 [ port <integer> ] [ dscp
        <integer> ] {
        <address_match_element>; ... };
    lmdb-mapsize <sizeval>;
    lock-file ( <quoted_string> | none );
    managed-keys-directory <quoted_string>;
    masterfile-format ( map | raw | text );
    masterfile-style ( full | relative );
    match-mapped-addresses <boolean>;
    max-cache-size ( default | unlimited | <sizeval> | <percentage> );
    max-cache-ttl <duration>;
    max-clients-per-query <integer>;
    max-journal-size ( default | unlimited | <sizeval> );
    max-ncache-ttl <duration>;
    max-records <integer>;
    max-recursion-depth <integer>;
```

(continues on next page)

(continued from previous page)

```
    max-recursion-queries <integer>;
    max-refresh-time <integer>;
    max-retry-time <integer>;
    max-rsa-exponent-size <integer>;
    max-stale-ttl <duration>;
    max-transfer-idle-in <integer>;
    max-transfer-idle-out <integer>;
    max-transfer-time-in <integer>;
    max-transfer-time-out <integer>;
    max-udp-size <integer>;
    max-zone-ttl ( unlimited | <duration> );
    memstatistics <boolean>;
    memstatistics-file <quoted_string>;
    message-compression <boolean>;
    min-cache-ttl <duration>;
    min-ncache-ttl <duration>;
    min-refresh-time <integer>;
    min-retry-time <integer>;
    minimal-any <boolean>;
    minimal-responses ( no-auth | no-auth-recursive | <boolean> );
    multi-master <boolean>;
    new-zones-directory <quoted_string>;
    no-case-compress { <address_match_element>; ... };
    nocookie-udp-size <integer>;
    notify ( explicit | master-only | <boolean> );
    notify-delay <integer>;
    notify-rate <integer>;
    notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
        dscp <integer> ];
    notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ]
        [ dscp <integer> ];
    notify-to-soa <boolean>;
    nta-lifetime <duration>;
    nta-recheck <duration>;
    nxdomain-redirect <string>;
    pid-file ( <quoted_string> | none );
    port <integer>;
    preferred-glue <string>;
    prefetch <integer> [ <integer> ];
    provide-ixfr <boolean>;
    qname-minimization ( strict | relaxed | disabled | off );
    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> ];
    querylog <boolean>;
    random-device ( <quoted_string> | none );
    rate-limit {
            all-per-second <integer>;
            errors-per-second <integer>;
```

(continues on next page)

(continued from previous page)

```
            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>;
    };
    recursing-file <quoted_string>;
    recursion <boolean>;
    recursive-clients <integer>;
    request-expire <boolean>;
    request-ixfr <boolean>;
    request-nsid <boolean>;
    require-server-cookie <boolean>;
    reserved-sockets <integer>;
    resolver-nonbackoff-tries <integer>;
    resolver-query-timeout <integer>;
    resolver-retry-interval <integer>;
    response-padding { <address_match_element>; ... } block-size
        <integer>;
    response-policy { zone <string> [ add-soa <boolean> ] [ log
        <boolean> ] [ max-policy-ttl <duration> ] [ min-update-interval
        <duration> ] [ policy ( cname | disabled | drop | given | no-op
        | nodata | nxdomain | passthru | tcp-only <quoted_string> ) ] [
        recursive-only <boolean> ] [ nsip-enable <boolean> ] [
        nsdname-enable <boolean> ]; ... } [ add-soa <boolean> ] [
        break-dnssec <boolean> ] [ max-policy-ttl <duration> ] [
        min-update-interval <duration> ] [ min-ns-dots <integer> ] [
        nsip-wait-recurse <boolean> ] [ qname-wait-recurse <boolean> ]
        [ recursive-only <boolean> ] [ nsip-enable <boolean> ] [
        nsdname-enable <boolean> ] [ dnsrps-enable <boolean> ] [
        dnsrps-options { <unspecified-text> } ];
    root-delegation-only [ exclude { <string>; ... } ];
    root-key-sentinel <boolean>;
    rrset-order { [ class <string> ] [ type <string> ] [ name
        <quoted_string> ] <string> <string>; ... };
    secroots-file <quoted_string>;
    send-cookie <boolean>;
    serial-query-rate <integer>;
    serial-update-method ( date | increment | unixtime );
    server-id ( <quoted_string> | none | hostname );
    servfail-ttl <duration>;
    session-keyalg <string>;
    session-keyfile ( <quoted_string> | none );
    session-keyname <string>;
```

(continues on next page)

(continued from previous page)

```
      sig-signing-nodes <integer>;
      sig-signing-signatures <integer>;
      sig-signing-type <integer>;
      sig-validity-interval <integer> [ <integer> ];
      sortlist { <address_match_element>; ... };
      stacksize ( default | unlimited | <sizeval> );
      stale-answer-enable <boolean>;
      stale-answer-ttl <duration>;
      stale-cache-enable <boolean>;
      startup-notify-rate <integer>;
      statistics-file <quoted_string>;
      synth-from-dnssec <boolean>;
      tcp-advertised-timeout <integer>;
      tcp-clients <integer>;
      tcp-idle-timeout <integer>;
      tcp-initial-timeout <integer>;
      tcp-keepalive-timeout <integer>;
      tcp-listen-queue <integer>;
      tkey-dhkey <quoted_string> <integer>;
      tkey-domain <quoted_string>;
      tkey-gssapi-credential <quoted_string>;
      tkey-gssapi-keytab <quoted_string>;
      transfer-format ( many-answers | one-answer );
      transfer-message-size <integer>;
      transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
          dscp <integer> ];
      transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * )
          ] [ dscp <integer> ];
      transfers-in <integer>;
      transfers-out <integer>;
      transfers-per-ns <integer>;
      trust-anchor-telemetry <boolean>; // experimental
      try-tcp-refresh <boolean>;
      update-check-ksk <boolean>;
      use-alt-transfer-source <boolean>;
      use-v4-udp-ports { <portrange>; ... };
      use-v6-udp-ports { <portrange>; ... };
      v6-bias <integer>;
      validate-except { <string>; ... };
      version ( <quoted_string> | none );
      zero-no-soa-ttl <boolean>;
      zero-no-soa-ttl-cache <boolean>;
      zone-statistics ( full | terse | none | <boolean> );
};
```

4.2.14 `options` Statement Definition and Usage

The `options` statement sets up global options to be used by BIND. This statement may appear only once in a configuration file. If there is no `options` statement, an options block with each option set to its default is used.

`attach-cache` This option 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 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 cache_name specifies the cache to be shared. When the `named` server configures views 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 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 of 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-accept-expired`, `dnssec-validation`, `max-cache-ttl`, `max-ncache-ttl`, `max-stale-ttl`, `max-cache-size`, `min-cache-ttl`, `min-ncache-ttl`, and `zero-no-soa-ttl`.

Note that 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 the administrator's responsibility to ensure configuration differences in different views do not cause disruption with a shared cache.

`directory` This sets the working directory of the server. Any non-absolute pathnames in the configuration file are taken as relative to this directory. The default location for most server output files (e.g., `named.run`) is this directory. If a directory is not specified, the working directory defaults to `"."`, the directory from which the server was started. The directory specified should be an absolute path, and *must* be writable by the effective user ID of the `named` process.

`dnstap` `dnstap` is a fast, flexible method for capturing and logging DNS traffic. Developed by Robert Edmonds at Farsight Security, Inc., and supported by multiple DNS implementations, `dnstap` uses `libfstrm` (a lightweight high-speed framing library; see https://github.com/farsightsec/fstrm) to send event payloads which are encoded using Protocol Buffers (`libprotobuf-c`, a mechanism for serializing structured data developed by Google, Inc.; see https://developers.google.com/protocol-buffers/).

To enable `dnstap` at compile time, the `fstrm` and `protobuf-c` libraries must be available, and BIND must be configured with `--enable-dnstap`.

The `dnstap` option is a bracketed list of message types to be logged. These may be set differently for each view. Supported types are `client`, `auth`, `resolver`, `forwarder`, and `update`. Specifying type `all` causes all `dnstap` messages to be logged, regardless of type.

Each type may take an additional argument to indicate whether to log `query` messages or `response` messages; if not specified, both queries and responses are logged.

Example: To log all authoritative queries and responses, recursive client responses, and upstream queries sent by the resolver, use:

```
dnstap {
  auth;
  client response;
  resolver query;
};
```

Logged `dnstap` messages can be parsed using the `dnstap-read` utility (see *dnstap-read - print dnstap data in human-readable form* for details).

For more information on `dnstap`, see http://dnstap.info.

The fstrm library has a number of tunables that are exposed in `named.conf`, and can be modified if necessary to improve performance or prevent loss of data. These are:

- `fstrm-set-buffer-hint`: The threshold number of bytes to accumulate in the output buffer before forcing a buffer flush. The minimum is 1024, the maximum is 65536, and the default is 8192.
- `fstrm-set-flush-timeout`: The number of seconds to allow unflushed data to remain in the output buffer. The minimum is 1 second, the maximum is 600 seconds (10 minutes), and the default is 1 second.
- `fstrm-set-output-notify-threshold`: The number of outstanding queue entries to allow on an input queue before waking the I/O thread. The minimum is 1 and the default is 32.
- `fstrm-set-output-queue-model`: The queuing semantics to use for queue objects. The default is `mpsc` (multiple producer, single consumer); the other option is `spsc` (single producer, single consumer).
- `fstrm-set-input-queue-size`: The number of queue entries to allocate for each input queue. This value must be a power of 2. The minimum is 2, the maximum is 16384, and the default is 512.
- `fstrm-set-output-queue-size`: The number of queue entries to allocate for each output queue. The minimum is 2, the maximum is system-dependent and based on `IOV_MAX`, and the default is 64.
- `fstrm-set-reopen-interval`: The number of seconds to wait between attempts to reopen a closed output stream. The minimum is 1 second, the maximum is 600 seconds (10 minutes), and the default is 5 seconds. For convenience, TTL-style time unit suffixes may be used to specify the value.

Note that all of the above minimum, maximum, and default values are set by the `libfstrm` library, and may be subject to change in future versions of the library. See the `libfstrm` documentation for more information.

`dnstap-output` This configures the path to which the `dnstap` frame stream is sent if `dnstap` is enabled at compile time and active.

The first argument is either `file` or `unix`, indicating whether the destination is a file or a Unix domain socket. The second argument is the path of the file or socket. (Note: when using a socket, `dnstap`

messages are only sent if another process such as `fstrm_capture` (provided with `libfstrm`) is listening on the socket.)

If the first argument is `file`, then up to three additional options can be added: `size` indicates the size to which a `dnstap` log file can grow before being rolled to a new file; `versions` specifies the number of rolled log files to retain; and `suffix` indicates whether to retain rolled log files with an incrementing counter as the suffix (`increment`) or with the current timestamp (`timestamp`). These are similar to the `size`, `versions`, and `suffix` options in a `logging` channel. The default is to allow `dnstap` log files to grow to any size without rolling.

`dnstap-output` can only be set globally in `options`. Currently, it can only be set once while `named` is running; once set, it cannot be changed by `rndc reload` or `rndc reconfig`.

`dnstap-identity` This specifies an `identity` string to send in `dnstap` messages. If set to `hostname`, which is the default, the server's hostname is sent. If set to `none`, no identity string is sent.

`dnstap-version` This specifies a `version` string to send in `dnstap` messages. The default is the version number of the BIND release. If set to `none`, no version string is sent.

`geoip-directory` When `named` is compiled using the MaxMind GeoIP2 geolocation API, this specifies the directory containing GeoIP database files. By default, the option is set based on the prefix used to build the `libmaxminddb` module; for example, if the library is installed in `/usr/local/lib`, then the default `geoip-directory` is `/usr/local/share/GeoIP`. On Windows, the default is the `named` working directory. See *acl Statement Definition and Usage* for details about `geoip` ACLs.

`key-directory` This is the directory where the public and private DNSSEC key files should be found when performing a dynamic update of secure zones, if different than the current working directory. (Note that this option has no effect on the paths for files containing non-DNSSEC keys such as `bind.keys`, `rndc.key`, or `session.key`.)

`lmdb-mapsize` When `named` is built with liblmdb, this option sets a maximum size for the memory map of the new-zone database (NZD) in LMDB database format. This database is used to store configuration information for zones added using `rndc addzone`. Note that this is not the NZD database file size, but the largest size that the database may grow to.

Because the database file is memory-mapped, its size is limited by the address space of the `named` process. The default of 32 megabytes was chosen to be usable with 32-bit `named` builds. The largest permitted value is 1 terabyte. Given typical zone configurations without elaborate ACLs, a 32 MB NZD file ought to be able to hold configurations of about 100,000 zones.

`managed-keys-directory` This specifies the directory in which to store the files that track managed DNSSEC keys (i.e., those configured using the `initial-key` or `initial-ds` keywords in a `trust-anchors` statement). By default, this is the working directory. The directory *must* be writable by the effective user ID of the `named` process.

If `named` is not configured to use views, managed keys for the server are tracked in a single file called `managed-keys.bind`. Otherwise, managed keys are tracked in separate files, one file per view; each file name is the view name (or, if it contains characters that are incompatible with use as a file name, the SHA256 hash of the view name), followed by the extension `.mkeys`.

(Note: in earlier releases, file names for views always used the SHA256 hash of the view name. To ensure compatibility after upgrading, if a file using the old name format is found to exist, it is used instead of the new format.)

`max-ixfr-ratio` This sets the size threshold (expressed as a percentage of the size of the full zone) beyond which `named` chooses to use an AXFR response rather than IXFR when answering zone transfer requests. See *Incremental Zone Transfers (IXFR)*.

`new-zones-directory` This specifies the directory in which to store the configuration parameters for zones added via `rndc addzone`. By default, this is the working directory. If set to a relative path, it is

relative to the working directory. The directory *must* be writable by the effective user ID of the `named` process.

`qname-minimization` This option controls QNAME minimization behavior in the BIND resolver. When set to `strict`, BIND follows the QNAME minimization algorithm to the letter, as specified in **RFC 7816**. Setting this option to `relaxed` causes BIND to fall back to normal (non-minimized) query mode when it receives either NXDOMAIN or other unexpected responses (e.g., SERVFAIL, improper zone cut, REFUSED) to a minimized query. `disabled` disables QNAME minimization completely. The current default is `relaxed`, but it may be changed to `strict` in a future release.

`tkey-gssapi-keytab` This is the KRB5 keytab file to use for GSS-TSIG updates. If this option is set and tkey-gssapi-credential is not set, updates are allowed with any key matching a principal in the specified keytab.

`tkey-gssapi-credential` This is the security credential with which the server should authenticate keys requested by the GSS-TSIG protocol. Currently only Kerberos 5 authentication is available; the credential is a Kerberos principal which the server can acquire through the default system key file, normally `/etc/krb5.keytab`. The location keytab file can be overridden using the tkey-gssapi-keytab option. Normally this principal is of the form `DNS/server.domain`. To use GSS-TSIG, `tkey-domain` must also be set if a specific keytab is not set with tkey-gssapi-keytab.

`tkey-domain` This domain is appended to the names of all shared keys generated with `TKEY`. 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 is `client specified part + tkey-domain`. Otherwise, the name of the shared key is `random hex digits + tkey-domain`. In most cases, the `domainname` should be the server's domain name, or an otherwise nonexistent subdomain like `_tkey.domainname`. If using GSS-TSIG, this variable must be defined, unless a specific keytab is specified using `tkey-gssapi-keytab`.

`tkey-dhkey` This is the Diffie-Hellman key used by the server to generate shared keys with clients using the Diffie-Hellman mode of `TKEY`. The server 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 server's host name.

`cache-file` This is for testing only. Do not use.

`dump-file` This is the pathname of the file the server dumps the database to, when instructed to do so with `rndc dumpdb`. If not specified, the default is `named_dump.db`.

`memstatistics-file` This is the pathname of the file the server writes memory usage statistics to on exit. If not specified, the default is `named.memstats`.

`lock-file` This is the pathname of a file on which `named` attempts to acquire a file lock when starting up for the first time; if unsuccessful, the server terminates, under the assumption that another server is already running. If not specified, the default is `none`.

Specifying `lock-file none` disables the use of a lock file. `lock-file` is ignored if `named` was run using the `-X` option, which overrides it. Changes to `lock-file` are ignored if `named` is being reloaded or reconfigured; it is only effective when the server is first started.

`pid-file` This is the pathname of the file the server writes its process ID in. If not specified, the default is `/var/run/named/named.pid`. The PID file is used by programs that send signals to the running name server. Specifying `pid-file none` disables the use of a PID file; no file is written and any existing one is removed. Note that `none` is a keyword, not a filename, and therefore is not enclosed in double quotes.

`recursing-file` This is the pathname of the file the server dumps the queries that are currently recursing, when instructed to do so with `rndc recursing`. If not specified, the default is `named.recursing`.

`statistics-file` This is the pathname of the file the server appends statistics to, when instructed to do so using `rndc stats`. If not specified, the default is `named.stats` in the server's current directory. The format of the file is described in *The Statistics File*.

bindkeys-file This is the pathname of a file to override the built-in trusted keys provided by named. See the discussion of dnssec-validation for details. If not specified, the default is /etc/bind.keys.

secroots-file This is the pathname of the file the server dumps security roots to, when instructed to do so with rndc secroots. If not specified, the default is named.secroots.

session-keyfile This is the pathname of the file into which to write a TSIG session key generated by named for use by nsupdate -l. If not specified, the default is /var/run/named/session.key. (See *Dynamic Update Policies*, and in particular the discussion of the update-policy statement's local option, for more information about this feature.)

session-keyname This is the key name to use for the TSIG session key. If not specified, the default is local-ddns.

session-keyalg This is 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. If not specified, the default is hmac-sha256.

port This is the UDP/TCP port number the server uses for receiving and sending DNS protocol traffic. The default is 53. This option is mainly intended for server testing; a server using a port other than 53 is not able to communicate with the global DNS.

dscp This is the global Differentiated Services Code Point (DSCP) value to classify outgoing DNS traffic, on operating systems that support DSCP. Valid values are 0 through 63. It is not configured by default.

random-device This specifies a source of entropy to be used by the server; it is a device or file from which to read entropy. If it is a file, operations requiring entropy will fail when the file has been exhausted.

Entropy is needed for cryptographic operations such as TKEY transactions, dynamic update of signed zones, and generation of TSIG session keys. It is also used for seeding and stirring the pseudo-random number generator which is used for less critical functions requiring randomness, such as generation of DNS message transaction IDs.

If random-device is not specified, or if it is set to none, entropy is read from the random number generation function supplied by the cryptographic library with which BIND was linked (i.e. OpenSSL or a PKCS#11 provider).

The random-device option takes effect during the initial configuration load at server startup time and is ignored on subsequent reloads.

preferred-glue If specified, the listed type (A or AAAA) is 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.

root-delegation-only This turns on enforcement of delegation-only in TLDs (top-level domains) and root zones with an optional exclude list.

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 server also serves a 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 whether they are signed by a child zone, and the authority section is 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 that some TLDs are not delegation-only; e.g., "DE", "LV", "US" and "MUSEUM". This list is not exhaustive.

```
options {
    root-delegation-only exclude { "de"; "lv"; "us"; "museum"; };
};
```

disable-algorithms This disables the specified DNSSEC algorithms at and below the specified name. Multiple `disable-algorithms` statements are allowed. Only the best match `disable-algorithms` clause is used to determine which algorithms are used.

If all supported algorithms are disabled, the zones covered by the `disable-algorithms` setting are treated as insecure.

Configured trust anchors in `trusted-anchors` (or `managed-keys` or `trusted-keys`) that match a disabled algorithm are ignored and treated as if they were not configured at all.

disable-ds-digests This disables the specified DS digest types at and below the specified name. Multiple `disable-ds-digests` statements are allowed. Only the best match `disable-ds-digests` clause is used to determine the digest types.

If all supported digest types are disabled, the zones covered by the `disable-ds-digests` are treated as insecure.

dnssec-must-be-secure This specifies hierarchies which must be or may not be secure (signed and validated). If `yes`, then `named` only accepts answers if they are secure. If `no`, then normal DNSSEC validation applies, allowing insecure answers to be accepted. The specified domain must be defined as a trust anchor, for instance in a `trust-anchors` statement, or `dnssec-validation auto` must be active.

dns64 This directive instructs `named` to return mapped IPv4 addresses to AAAA queries when there are no AAAA records. It is intended to be used in conjunction with a NAT64. Each `dns64` defines one DNS64 prefix. Multiple DNS64 prefixes can be defined.

Compatible IPv6 prefixes have lengths of 32, 40, 48, 56, 64, and 96, per **RFC 6052**. Bits 64..71 inclusive must be zero with the most significant bit of the prefix in position 0.

In addition, a reverse IP6.ARPA zone is created for the prefix to provide a mapping from the IP6.ARPA names to the corresponding IN-ADDR.ARPA names using synthesized CNAMEs. `dns64-server` and `dns64-contact` can be used to specify the name of the server and contact for the zones. These can be set at the view/options level but not on a per-prefix basis.

Each `dns64` supports an optional `clients` ACL that determines which clients are affected by this directive. If not defined, it defaults to `any;`.

Each `dns64` supports an optional `mapped` ACL that selects which IPv4 addresses are to be mapped in the corresponding A RRset. If not defined, it defaults to `any;`.

Normally, DNS64 does not apply to a domain name that owns one or more AAAA records; these records are simply returned. The optional `exclude` ACL allows specification of a list of IPv6 addresses that are ignored if they appear in a domain name's AAAA records; DNS64 is applied to any A records the domain name owns. If not defined, `exclude` defaults to ::ffff:0.0.0.0/96.

An optional `suffix` can also be defined to set the bits trailing the mapped IPv4 address bits. By default these bits are set to `::`. The bits matching the prefix and mapped IPv4 address must be zero.

If `recursive-only` is set to `yes` the DNS64 synthesis only happens for recursive queries. The default is `no`.

If `break-dnssec` is set to `yes` the DNS64 synthesis happens even if the result, if validated, would cause a DNSSEC validation failure. If this option is set to `no` (the default), the DO is set on the incoming

query, and there are RRSIGs on the applicable records, then synthesis does not happen.

```
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 ::;
};
```

dnssec-loadkeys-interval When a zone is configured with `auto-dnssec 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: DNSSEC key generation tool* and *dnssec-settime: set the key timing metadata for a DNSSEC key*). The `dnssec-loadkeys-interval` option sets the frequency of automatic repository checks, in minutes. The default is `60` (1 hour), the minimum is `1` (1 minute), and the maximum is `1440` (24 hours); any higher value is silently reduced.

dnssec-policy This specifies which key and signing policy (KASP) should be used for this zone. This is a string referring to a `dnssec-policy` statement. There are two built-in policies: `default`, which uses the default policy, and `none`, which means no DNSSEC policy and keeps the zone unsigned. The default is `none`. See *dnssec-policy Grammar* for more details.

dnssec-update-mode If this option is set to its default value of `maintain` in a zone of type `master` which is DNSSEC-signed and configured to allow dynamic updates (see *Dynamic Update Policies*), and if `named` has access to the private signing key(s) for the zone, then `named` automatically signs all new or changed records and maintains signatures for the zone by regenerating RRSIG records whenever they approach their expiration date.

If the option is changed to `no-resign`, then `named` signs all new or changed records, but scheduled maintenance of signatures is disabled.

With either of these settings, `named` rejects updates to a DNSSEC-signed zone when the signing keys are inactive or unavailable to `named`. (A planned third option, `external`, will disable all automatic signing and allow DNSSEC data to be submitted into a zone via dynamic update; this is not yet implemented.)

nta-lifetime This specifies the default lifetime, in seconds, for negative trust anchors added via `rndc nta`.

A negative trust anchor selectively disables DNSSEC validation for zones that are known to be failing because of misconfiguration, rather than an attack. When data to be validated is at or below an active NTA (and above any other configured trust anchors), `named` aborts the DNSSEC validation process and treats the data as insecure rather than bogus. This continues until the NTA's lifetime has elapsed. NTAs persist across `named` restarts.

For convenience, TTL-style time unit suffixes can be used to specify the NTA lifetime in seconds, minutes, or hours. It also accepts ISO 8601 duration formats.

`nta-lifetime` defaults to one hour; it cannot exceed one week.

nta-recheck This specifies how often to check whether negative trust anchors added via `rndc nta` are still necessary.

A negative trust anchor is normally used when a domain has stopped validating due to operator error; it temporarily disables DNSSEC validation for that domain. In the interest of ensuring that DNSSEC validation is turned back on as soon as possible, `named` periodically sends a query to the domain, ignoring negative trust anchors, to find out whether it can now be validated. If so, the negative trust anchor is allowed to expire early.

Validity checks can be disabled for an individual NTA by using `rndc nta -f`, or for all NTAs by setting `nta-recheck` to zero.

For convenience, TTL-style time unit suffixes can be used to specify the NTA recheck interval in seconds, minutes, or hours. It also accepts ISO 8601 duration formats.

The default is five minutes. It cannot be longer than `nta-lifetime`, which cannot be longer than a week.

`max-zone-ttl` This specifies a maximum permissible TTL value in seconds. For convenience, TTL-style time unit suffixes may be used to specify the maximum value. When loading a zone file using a `masterfile-format` of `text` or `raw`, any record encountered with a TTL higher than `max-zone-ttl` causes 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. The `max-zone-ttl` option guarantees that the largest TTL in the zone is no higher than the set value.

(NOTE: Because `map`-format files load directly into memory, this option cannot be used with them.)

The default value is `unlimited`. A `max-zone-ttl` of zero is treated as `unlimited`.

`stale-answer-ttl` This specifies the TTL to be returned on stale answers. The default is 1 second. The minimum allowed is also 1 second; a value of 0 is updated silently to 1 second.

For stale answers to be returned, they must be enabled, either in the configuration file using `stale-answer-enable` or via `rndc serve-stale on`.

`serial-update-method` Zones configured for dynamic DNS may use this option to set the update method to be used for the zone serial number in the SOA record.

With the default setting of `serial-update-method increment;`, the SOA serial number is incremented by one each time the zone is updated.

When set to `serial-update-method unixtime;`, the SOA serial number is 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 simply incremented by one.

When set to `serial-update-method date;`, the new SOA serial number is the current date in the form "YYYYMMDD", followed by two zeroes, unless the existing serial number is already greater than or equal to that value, in which case it is incremented by one.

`zone-statistics` If `full`, the server collects statistical data on all zones, unless specifically turned off on a per-zone basis by specifying `zone-statistics terse` or `zone-statistics none` in the `zone` statement. The default is `terse`, providing minimal statistics on zones (including name and current serial number, but not query type counters).

These statistics may be accessed via the `statistics-channel` or using `rndc stats`, which dumps them to the file listed in the `statistics-file`. See also *The Statistics File*.

For backward compatibility with earlier versions of BIND 9, the `zone-statistics` option can also accept `yes` or `no`; `yes` has the same meaning as `full`. As of BIND 9.10, `no` has the same meaning as `none`; previously, it was the same as `terse`.

Boolean Options

`automatic-interface-scan` If `yes` and supported by the operating system, this automatically rescans network interfaces when the interface addresses are added or removed. The default is `yes`. This configuration option does not affect the time-based `interface-interval` option; it is recommended to set the time-based `interface-interval` to 0 when the operator confirms that automatic interface scanning is supported by the operating system.

The `automatic-interface-scan` implementation uses routing sockets for the network interface discovery; therefore, the operating system has to support the routing sockets for this feature to work.

`allow-new-zones` If `yes`, then zones can be added at runtime via `rndc addzone`. The default is `no`.

Newly added zones' configuration parameters are stored so that they can persist after the server is restarted. The configuration information is saved in a file called `viewname.nzf` (or, if `named` is compiled with liblmdb, in an LMDB database file called `viewname.nzd`). "viewname" is the name of the view, unless the view name contains characters that are incompatible with use as a file name, in which case a cryptographic hash of the view name is used instead.

Configurations for zones added at runtime are stored either in a new-zone file (NZF) or a new-zone database (NZD), depending on whether `named` was linked with liblmdb at compile time. See *rndc - name server control utility* for further details about `rndc addzone`.

`auth-nxdomain` If `yes`, then the `AA` bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default is `no`. If using very old DNS software, this may need to be set to `yes`.

`deallocate-on-exit` This option was used in BIND 8 to enable checking for memory leaks on exit. BIND 9 ignores the option and always performs the checks.

`memstatistics` This writes memory statistics to the file specified by `memstatistics-file` at exit. The default is `no` unless '-m record' is specified on the command line, in which case it is `yes`.

`dialup` If `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. Although this setting has different effects according to zone type, it concentrates the zone maintenance so that everything happens quickly, once every `heartbeat-interval`, ideally during a single call. It also suppresses some normal zone maintenance traffic. The default is `no`.

If specified in the `view` and `zone` statements, the `dialup` option overrides the global `dialup` option.

If the zone is a primary zone, the server sends out a NOTIFY request to all the secondaries (default). This should trigger the zone serial number check in the secondary (providing it supports NOTIFY), allowing the secondary to verify the zone while the connection is active. The set of servers to which NOTIFY is sent can be controlled by `notify` and `also-notify`.

If the zone is a secondary or stub zone, the server suppresses the regular "zone up to date" (refresh) queries and only performs them when the `heartbeat-interval` expires, in addition to sending NOTIFY requests.

Finer control can be achieved by using `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.

dialup mode	normal refresh	heart-beat refresh	heart-beat 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 that normal NOTIFY processing is not affected by `dialup`.

`flush-zones-on-shutdown` When the name server exits upon receiving SIGTERM, flush or do not flush any pending zone writes. The default is `flush-zones-on-shutdown no`.

`geoip-use-ecs` This option was part of an experimental implementation of the EDNS CLIENT-SUBNET for authoritative servers, but is now obsolete.

`root-key-sentinel` If `yes`, respond to root key sentinel probes as described in draft-ietf-dnsop-kskroll-sentinel-08. The default is `yes`.

`message-compression` If `yes`, DNS name compression is used in responses to regular queries (not including AXFR or IXFR, which always use compression). Setting this option to `no` reduces CPU usage on servers and may improve throughput. However, it increases response size, which may cause more queries to be processed using TCP; a server with compression disabled is out of compliance with **RFC 1123** Section 6.1.3.2. The default is `yes`.

`minimal-responses` This option controls the addition of records to the authority and additional sections of responses. Such records may be included in responses to be helpful to clients; for example, NS or MX records may have associated address records included in the additional section, obviating the need for a separate address lookup. However, adding these records to responses is not mandatory and requires additional database lookups, causing extra latency when marshalling responses. `minimal-responses` takes one of four values:

- `no`: the server is as complete as possible when generating responses.
- `yes`: the server only adds records to the authority and additional sections when such records are required by the DNS protocol (for example, when returning delegations or negative responses). This provides the best server performance but may result in more client queries.
- `no-auth`: the server omits records from the authority section except when they are required, but it may still add records to the additional section.
- `no-auth-recursive`: the same as `no-auth` when recursion is requested in the query (RD=1), or the same as `no` if recursion is not requested.

`no-auth` and `no-auth-recursive` are useful when answering stub clients, which usually ignore the authority section. `no-auth-recursive` is meant for use in mixed-mode servers that handle both authoritative and recursive queries.

The default is `no-auth-recursive`.

`glue-cache` When set to `yes`, a cache is used to improve query performance when adding address-type (A and AAAA) glue records to the additional section of DNS response messages that delegate to a child zone.

The glue cache uses memory proportional to the number of delegations in the zone. The default setting is `yes`, which improves performance at the cost of increased memory usage for the zone. To avoid this, set it to `no`.

`minimal-any` If set to `yes`, the server replies with only one of the RRsets for the query name when generating a positive response to a query of type ANY over UDP, and its covering RRSIGs if any, instead of replying with all known RRsets for the name. Similarly, a query for type RRSIG is answered with the RRSIG records covering only one type. This can reduce the impact of some kinds of attack traffic, without harming legitimate clients. (Note, however, that the RRset returned is the first one found in the database; it is not necessarily the smallest available RRset.) Additionally, `minimal-responses` is turned on for these queries, so no unnecessary records are added to the authority or additional sections. The default is `no`.

`notify` If `yes` (the default), DNS NOTIFY messages are sent when a zone the server is authoritative for changes; see *Notify*. The messages are sent to the servers listed in the zone's NS records (except the primary server identified in the SOA MNAME field), and to any servers listed in the `also-notify` option.

If `master-only`, notifies are only sent for primary zones. If `explicit`, notifies are sent only to servers explicitly listed using `also-notify`. If `no`, no notifies are sent.

The `notify` option may also be specified in the `zone` statement, in which case it overrides the `options notify` statement. It would only be necessary to turn off this option if it caused secondary zones to crash.

`notify-to-soa` If `yes`, do not check the name servers in the NS RRset against the SOA MNAME. Normally a NOTIFY message is not sent to the SOA MNAME (SOA ORIGIN), as it is supposed to contain the name of the ultimate primary server. Sometimes, however, a secondary server is listed as the SOA MNAME in hidden primary configurations; in that case, the ultimate primary should be set to still send NOTIFY messages to all the name servers listed in the NS RRset.

`recursion` If `yes`, and a DNS query requests recursion, then the server attempts to do all the work required to answer the query. If recursion is off and the server does not already know the answer, it returns a referral response. The default is `yes`. Note that setting `recursion no` does not prevent clients from getting data from the server's cache; it only prevents new data from being cached as an effect of client queries. Caching may still occur as an effect of the server's internal operation, such as NOTIFY address lookups.

`request-nsid` If `yes`, then an empty EDNS(0) NSID (Name Server Identifier) option is sent with all queries to authoritative name servers during iterative resolution. If the authoritative server returns an NSID option in its response, then its contents are logged in the `nsid` category at level `info`. The default is `no`.

`request-sit` This experimental option is obsolete.

`require-server-cookie` If `yes`, require a valid server cookie before sending a full response to a UDP request from a cookie-aware client. BADCOOKIE is sent if there is a bad or nonexistent server cookie.

The default is `no`.

Users wishing to test that DNS COOKIE clients correctly handle BADCOOKIE, or who are getting a lot of forged DNS requests with DNS COOKIES present, should set this to `yes`. Setting this to `yes` results in a reduced amplification effect in a reflection attack, as the BADCOOKIE response is smaller than a full response, while also requiring a legitimate client to follow up with a second query with the new, valid, cookie.

`answer-cookie` When set to the default value of `yes`, COOKIE EDNS options are sent when applicable in replies to client queries. If set to `no`, COOKIE EDNS options are not sent in replies. This can only be set at the global options level, not per-view.

`answer-cookie no` is intended as a temporary measure, for use when `named` shares an IP address with other servers that do not yet support DNS COOKIE. A mismatch between servers on the same address is not expected to cause operational problems, but the option to disable COOKIE responses so that all servers have the same behavior is provided out of an abundance of caution. DNS COOKIE is an important security mechanism, and should not be disabled unless absolutely necessary.

`send-cookie` If `yes`, then a COOKIE EDNS option is sent along with the query. If the resolver has previously communicated with the server, the COOKIE returned in the previous transaction is sent. This is used by the server to determine whether the resolver has talked to it before. A resolver sending the correct COOKIE is 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 option are not subject to response rate limiting (RRL). Resolvers which do not send a correct COOKIE option may be limited to receiving smaller responses via the `nocookie-udp-size` option.

The default is `yes`.

`stale-answer-enable` If `yes`, enable the returning of "stale" cached answers when the name servers for a zone are not answering and the `stale-cache-enable` option is also enabled. The default is not to return stale answers.

Stale answers can also be enabled or disabled at runtime via `rndc serve-stale on` or `rndc`

`serve-stale off`; these override the configured setting. `rndc serve-stale reset` restores the setting to the one specified in `named.conf`. Note that if stale answers have been disabled by `rndc`, they cannot be re-enabled by reloading or reconfiguring `named`; they must be re-enabled with `rndc serve-stale on`, or the server must be restarted.

Information about stale answers is logged under the `serve-stale` log category.

`stale-cache-enable` If `yes`, enable the retaining of "stale" cached answers. Default `yes`.

`nocookie-udp-size` This sets the maximum size of UDP responses that are sent to queries without a valid server COOKIE. A value below 128 is silently raised to 128. The default value is 4096, but the `max-udp-size` option may further limit the response size.

`sit-secret` This experimental option is obsolete.

`cookie-algorithm` This sets the algorithm to be used when generating the server cookie; the options are "aes", "sha1", or "sha256". The default is "aes" if supported by the cryptographic library; otherwise, "sha256".

`cookie-secret` If set, this is a shared secret used for generating and verifying EDNS COOKIE options within an anycast cluster. If not set, the system generates a random secret at startup. The shared secret is encoded as a hex string and needs to be 128 bits for AES128, 160 bits for SHA1, and 256 bits for SHA256.

If there are multiple secrets specified, the first one listed in `named.conf` is used to generate new server cookies. The others are only used to verify returned cookies.

`response-padding` The EDNS Padding option is intended to improve confidentiality when DNS queries are sent over an encrypted channel by reducing the variability in packet sizes. If a query:

1. contains an EDNS Padding option,
2. includes a valid server cookie or uses TCP,
3. is not signed using TSIG or SIG(0), and
4. is from a client whose address matches the specified ACL,

then the response is padded with an EDNS Padding option to a multiple of `block-size` bytes. If these conditions are not met, the response is not padded.

If `block-size` is 0 or the ACL is `none;`, this feature is disabled and no padding occurs; this is the default. If `block-size` is greater than 512, a warning is logged and the value is truncated to 512. Block sizes are ordinarily expected to be powers of two (for instance, 128), but this is not mandatory.

`trust-anchor-telemetry` This causes `named` to send specially formed queries once per day to domains for which trust anchors have been configured via, e.g., `dnssec-keys` or `dnssec-validation auto`.

The query name used for these queries has the form "_ta-xxxx(-xxxx)(…)".<domain>, where each "xxxx" is a group of four hexadecimal digits representing the key ID of a trusted DNSSEC key. The key IDs for each domain are sorted smallest to largest prior to encoding. The query type is NULL.

By monitoring these queries, zone operators are able to see which resolvers have been updated to trust a new key; this may help them decide when it is safe to remove an old one.

The default is `yes`.

`use-ixfr` *This option is obsolete.* To disable IXFR to a particular server or servers, see the information on the `provide-ixfr` option in *server Statement Definition and Usage*. See also *Incremental Zone Transfers (IXFR)*.

`provide-ixfr` See the description of `provide-ixfr` in *server Statement Definition and Usage*.

`request-ixfr` See the description of `request-ixfr` in *server Statement Definition and Usage*.

request-expire See the description of request-expire in *server Statement Definition and Usage*.

match-mapped-addresses If yes, then an IPv4-mapped IPv6 address matches any address match list entries that match the corresponding IPv4 address.

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. However, named now solves this problem internally. The use of this option is discouraged.

ixfr-from-differences When yes and the server loads a new version of a primary zone from its zone file or receives a new version of a secondary file via zone transfer, it compares the new version to the previous one and calculates a set of differences. The differences are then logged in the zone's journal file so that the changes can be transmitted to downstream secondaries as an incremental zone transfer.

By allowing incremental zone transfers to be used for non-dynamic zones, this option saves bandwidth at the expense of increased CPU and memory consumption at the primary server. In particular, if the new version of a zone is completely different from the previous one, the set of differences is of a size comparable to the combined size of the old and new zone versions, and the server needs to temporarily allocate memory to hold this complete difference set.

ixfr-from-differences also accepts master (or primary) and slave (or secondary) at the view and options levels, which causes ixfr-from-differences to be enabled for all primary or secondary zones, respectively. It is off for all zones by default.

Note: if inline signing is enabled for a zone, the user-provided ixfr-from-differences setting is ignored for that zone.

multi-master This should be set when there are multiple primary servers for a zone and the addresses refer to different machines. If yes, named does not log when the serial number on the primary is less than what named currently has. The default is no.

auto-dnssec Zones configured for dynamic DNS may use this option to allow varying levels of automatic DNSSEC key management. There are three possible settings:

auto-dnssec allow; permits keys to be updated and the zone fully re-signed whenever the user issues the command rndc sign zonename.

auto-dnssec maintain; includes the above, but also automatically adjusts the zone's DNSSEC keys on a schedule, according to the keys' timing metadata (see *dnssec-keygen: DNSSEC key generation tool* and *dnssec-settime: set the key timing metadata for a DNSSEC key*). The command rndc sign zonename causes named to load keys from the key repository and sign the zone with all keys that are active. rndc loadkeys zonename 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 is searched for changes periodically, regardless of whether rndc loadkeys is used. The recheck interval is defined by dnssec-loadkeys-interval.

The default setting is auto-dnssec off.

dnssec-enable This option is obsolete and has no effect.

dnssec-validation This option enables DNSSEC validation in named.

If set to auto, DNSSEC validation is enabled and a default trust anchor for the DNS root zone is used.

If set to yes, DNSSEC validation is enabled, but a trust anchor must be manually configured using a trust-anchors statement (or the managed-keys or trusted-keys statements, both deprecated). If there is no configured trust anchor, validation does not take place.

If set to no, DNSSEC validation is disabled.

The default is `auto`, unless BIND is built with `configure --disable-auto-validation`, in which case the default is `yes`.

The default root trust anchor is stored in the file `bind.keys`. `named` loads that key at startup if `dnssec-validation` is set to `auto`. A copy of the file is installed along with BIND 9, and is current as of the release date. If the root key expires, a new copy of `bind.keys` can be downloaded from https://www.isc.org/bind-keys.

(To prevent problems if `bind.keys` is not found, the current trust anchor is also compiled in `named`. Relying on this is not recommended, however, as it requires `named` to be recompiled with a new key when the root key expires.)

Note: `named` loads *only* the root key from `bind.keys`. The file cannot be used to store keys for other zones. The root key in `bind.keys` is ignored if `dnssec-validation auto` is not in use.

Whenever the resolver sends out queries to an EDNS-compliant server, it always sets the DO bit indicating it can support DNSSEC responses, even if `dnssec-validation` is off.

`validate-except` This specifies a list of domain names at and beneath which DNSSEC validation should *not* be performed, regardless of the presence of a trust anchor at or above those names. This may be used, for example, when configuring a top-level domain intended only for local use, so that the lack of a secure delegation for that domain in the root zone does not cause validation failures. (This is similar to setting a negative trust anchor except that it is a permanent configuration, whereas negative trust anchors expire and are removed after a set period of time.)

`dnssec-accept-expired` This accepts expired signatures when verifying DNSSEC signatures. The default is `no`. Setting this option to `yes` leaves `named` vulnerable to replay attacks.

`querylog` Query logging provides a complete log of all incoming queries and all query errors. This provides more insight into the server's activity, but with a cost to performance which may be significant on heavily-loaded servers.

The `querylog` option specifies whether query logging should be active when `named` first starts. If `querylog` is not specified, then query logging is determined by the presence of the logging category `queries`. Query logging can also be activated at runtime using the command `rndc querylog on`, or deactivated with `rndc querylog off`.

`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 usage area. For `primary` zones the default is `fail`. For `secondary` 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 indicates that it is a reverse lookup of a hostname (the owner name ends in IN-ADDR.ARPA, IP6.ARPA, or IP6.INT).

`check-dup-records` This checks primary zones for records that are treated as different by DNSSEC but are semantically equal in plain DNS. The default is to `warn`. Other possible values are `fail` and `ignore`.

`check-mx` This checks whether the MX record appears to refer to an IP address. The default is to `warn`. Other possible values are `fail` and `ignore`.

`check-wildcard` This option is used to check for non-terminal wildcards. The use of non-terminal wildcards is almost always as a result of a lack of understanding of the wildcard matching algorithm (**RFC 1034**).

This option affects primary zones. The default (`yes`) is to check for non-terminal wildcards and issue a warning.

`check-integrity` This performs post-load zone integrity checks on primary zones. It 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 is `yes`.

The use of the SPF record to publish Sender Policy Framework is deprecated, as the migration from using TXT records to SPF records was abandoned. Enabling this option also checks that a TXT Sender Policy Framework record exists (starts with "v=spf1") if there is an SPF record. Warnings are emitted if the TXT record does not exist; they can be suppressed with `check-spf`.

`check-mx-cname` If `check-integrity` is set, then fail, warn, or ignore MX records that refer to CNAMES. The default is to `warn`.

`check-srv-cname` If `check-integrity` is set, then fail, warn, or ignore SRV records that refer to CNAMES. The default is to `warn`.

`check-sibling` When performing integrity checks, also check that sibling glue exists. The default is `yes`.

`check-spf` If `check-integrity` is set, check that there is a TXT Sender Policy Framework record present (starts with "v=spf1") if there is an SPF record present. The default is `warn`.

`zero-no-soa-ttl` If `yes`, when returning authoritative negative responses to SOA queries, set the TTL of the SOA record returned in the authority section to zero. The default is `yes`.

`zero-no-soa-ttl-cache` If `yes`, when caching a negative response to a SOA query set the TTL to zero. The default is `no`.

`update-check-ksk` When set to the default value of `yes`, check the KSK bit in each key to determine how the key should be used when generating RRSIGs for a secure zone.

Ordinarily, zone-signing keys (that is, keys without the KSK bit set) are used to sign the entire zone, while key-signing keys (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 the `dnssec-signzone -z` command line option.

When this option is set to `yes`, there must be at least two active keys for every algorithm represented in the DNSKEY RRset: at least one KSK and one ZSK per algorithm. If there is any algorithm for which this requirement is not met, this option is ignored for that algorithm.

`dnssec-dnskey-kskonly` When this option and `update-check-ksk` are both set to `yes`, only key-signing keys (that is, keys with the KSK bit set) are used to sign the DNSKEY, CDNSKEY, and CDS RRsets at the zone apex. Zone-signing keys (keys without the KSK bit set) are used to sign the remainder of the zone, but not the DNSKEY RRset. This is similar to the `dnssec-signzone -x` command line option.

The default is `no`. If `update-check-ksk` is set to `no`, this option is ignored.

`try-tcp-refresh` If `yes`, try to refresh the zone using TCP if UDP queries fail. The default is `yes`.

`dnssec-secure-to-insecure` This allows a dynamic zone to transition from secure to insecure (i.e., signed to unsigned) by deleting all of the DNSKEY records. The default is `no`. If set to `yes`, and if the DNSKEY RRset at the zone apex is deleted, all RRSIG and NSEC records are removed from the zone as well.

If the zone uses NSEC3, it is also necessary to delete the NSEC3PARAM RRset from the zone apex; this causes the removal of all corresponding NSEC3 records. (It is expected that this requirement will be eliminated in a future release.)

Note that if a zone has been configured with `auto-dnssec maintain` and the private keys remain accessible in the key repository, the zone will be automatically signed again the next time `named` is started.

`synth-from-dnssec` This option synthesizes answers from cached NSEC, NSEC3, and other RRsets that have been proved to be correct using DNSSEC. The default is `no`, but it will become `yes` again in future releases.

Note: DNSSEC validation must be enabled for this option to be effective. This initial implementation only covers synthesis of answers from NSEC records; synthesis from NSEC3 is planned for the future. This will also be controlled by `synth-from-dnssec`.

Forwarding

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` This option is only meaningful if the forwarders list is not empty. A value of `first` is the default and causes the server to query the forwarders first; if that does not answer the question, the server then looks for the answer itself. If `only` is specified, the server only queries the forwarders.

`forwarders` This specifies a list of IP addresses to which queries are forwarded. The default is the empty list (no forwarding). Each address in the list can be associated with an optional port number and/or DSCP value, and a default port number and DSCP value can be set for the entire list.

Forwarding can also be configured on a per-domain basis, allowing for the global forwarding options to be overridden in a variety of ways. Particular domains can be set to use different forwarders, or have a different `forward only/first` behavior, or not forward at all; see *zone Statement Grammar*.

Dual-stack Servers

Dual-stack servers are used as servers of last resort, to work around problems in reachability due to the lack of support for either IPv4 or IPv6 on the host machine.

`dual-stack-servers` This specifies host names or addresses of machines with access to both IPv4 and IPv6 transports. If a hostname is used, the server must be able to resolve the name using only the transport it has. If the machine is dual-stacked, the `dual-stack-servers` parameter has no effect unless access to a transport has been disabled on the command line (e.g., `named -4`).

Access Control

Access to the server can be restricted based on the IP address of the requesting system. See *Address Match Lists* for details on how to specify IP address lists.

`allow-notify` This ACL specifies which hosts may send NOTIFY messages to inform this server of changes to zones for which it is acting as a secondary server. This is only applicable for secondary zones (i.e., type `secondary` or `slave`).

If this option is set in `view` or `options`, it is globally applied to all secondary zones. If set in the `zone` statement, the global value is overridden.

If not specified, the default is to process NOTIFY messages only from the configured `masters` for the zone. `allow-notify` can be used to expand the list of permitted hosts, not to reduce it.

`allow-query` This specifies which hosts are allowed to ask ordinary DNS questions. `allow-query` may also be specified in the `zone` statement, in which case it overrides the `options allow-query` statement. If not specified, the default is to allow queries from all hosts.

Note: `allow-query-cache` is used to specify access to the cache.

`allow-query-on` This specifies which local addresses can accept ordinary DNS questions. This makes it possible, for instance, to allow queries on internal-facing interfaces but disallow them on external-facing ones, without necessarily knowing the internal network's addresses.

Note that `allow-query-on` is only checked for queries that are permitted by `allow-query`. A query must be allowed by both ACLs, or it is refused.

`allow-query-on` may also be specified in the `zone` statement, in which case it overrides the `options allow-query-on` statement.

If not specified, the default is to allow queries on all addresses.

Note: `allow-query-cache` is used to specify access to the cache.

`allow-query-cache` This specifies which hosts are allowed to get answers from the cache. If `allow-recursion` is not set, BIND checks to see if the following parameters are set, in order: `allow-query-cache` and `allow-query` (unless `recursion no;` is set). If neither of those parameters is set, the default (localnets; localhost;) is used.

`allow-query-cache-on` This specifies which local addresses can send answers from the cache. If `allow-query-cache-on` is not set, then `allow-recursion-on` is used if set. Otherwise, the default is to allow cache responses to be sent from any address. Note: Both `allow-query-cache` and `allow-query-cache-on` must be satisfied before a cache response can be sent; a client that is blocked by one cannot be allowed by the other.

`allow-recursion` This specifies which hosts are allowed to make recursive queries through this server. BIND checks to see if the following parameters are set, in order: allow-query-cache and allow-query. If neither of those parameters is set, the default (localnets; localhost;) is used.

`allow-recursion-on` This specifies which local addresses can accept recursive queries. If `allow-recursion-on` is not set, then `allow-query-cache-on` is used if set; otherwise, the default is to allow recursive queries on all addresses. Any client permitted to send recursive queries can send them to any address on which `named` is listening. Note: Both `allow-recursion` and `allow-recursion-on` must be satisfied before recursion is allowed; a client that is blocked by one cannot be allowed by the other.

`allow-update` When set in the `zone` statement for a primary zone, this specifies which hosts are allowed to submit Dynamic DNS updates to that zone. The default is to deny updates from all hosts.

Note that allowing updates based on the requestor's IP address is insecure; see *Dynamic Update Security* for details.

In general this option should only be set at the `zone` level. While a default value can be set at the `options` or `view` level and inherited by zones, this could lead to some zones unintentionally allowing updates.

`allow-update-forwarding` When set in the `zone` statement for a secondary zone, this specifies which hosts are allowed to submit Dynamic DNS updates and have them be forwarded to the primary. The default

is `{ none; }`, which means that no update forwarding is performed.

To enable update forwarding, specify `allow-update-forwarding { any; };` in the `zone` statement. Specifying values other than `{ none; }` or `{ any; }` is usually counterproductive; the responsibility for update access control should rest with the primary server, not the secondary.

Note that enabling the update forwarding feature on a secondary server may expose primary servers to attacks if they rely on insecure IP-address-based access control; see *Dynamic Update Security* for more details.

In general this option should only be set at the `zone` level. While a default value can be set at the `options` or `view` level and inherited by zones, this can lead to some zones unintentionally forwarding updates.

`allow-v6-synthesis` This option was introduced for the smooth transition from AAAA to A6 and from "nibble labels" to binary labels. However, since both A6 and binary labels were then deprecated, this option was also deprecated. It is now ignored with some warning messages.

`allow-transfer` This specifies which hosts are allowed to receive zone transfers from the server. `allow-transfer` may also be specified in the `zone` statement, in which case it overrides the `allow-transfer` statement set in `options` or `view`. If not specified, the default is to allow transfers to all hosts.

`blackhole` This specifies a list of addresses which the server does not accept queries from or use to resolve a query. Queries from these addresses are not be responded to. The default is `none`.

`keep-response-order` This specifies a list of addresses to which the server sends responses to TCP queries, in the same order in which they were received. This disables the processing of TCP queries in parallel. The default is `none`.

`no-case-compress` This specifies a list of addresses which require responses to use case-insensitive compression. This ACL can be used when `named` needs to work with clients that do not comply with the requirement in **RFC 1034** to use case-insensitive name comparisons when checking for matching domain names.

If left undefined, the ACL defaults to `none`: case-insensitive compression is used for all clients. If the ACL is defined and matches a client, then case is ignored when compressing domain names in DNS responses sent to that client.

This can result in slightly smaller responses; if a response contains the names "example.com" and "example.COM", case-insensitive compression treats the second one as a duplicate. It also ensures that the case of the query name exactly matches the case of the owner names of returned records, rather than matching the case of the records entered in the zone file. This allows responses to exactly match the query, which is required by some clients due to incorrect use of case-sensitive comparisons.

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` does not preserve the case of owner names of records: if a zone file defines 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 use the *first* version of the name that was used in the zone file. This limitation may be addressed in a future release. However, domain names specified in the rdata of resource records (i.e., records of type NS, MX, CNAME, etc) always have their case preserved unless the client matches this ACL.

`resolver-query-timeout` This is the amount of time in milliseconds that the resolver spends attempting to resolve a recursive query before failing. The default and minimum is `10000` and the maximum is `30000`. Setting it to `0` results in the default being used.

This value was originally specified in seconds. Values less than or equal to 300 are treated as seconds and converted to milliseconds before applying the above limits.

Interfaces

The interfaces and ports that the server answers 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 listens on all interfaces allowed by the address match list. If a port is not specified, port 53 is 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; };
```

enables 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 listens 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 listens for incoming queries sent using IPv6. If not specified, the server listens on port 53 on all IPv6 interfaces.

Multiple `listen-on-v6` options can be used. For example,

```
listen-on-v6 { any; };
listen-on-v6 port 1234 { !2001:db8::/32; any; };
```

enables the name server on port 53 for any IPv6 addresses (with a single wildcard socket), and on port 1234 of IPv6 addresses that are 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; };
```

Query Address

If the server does not know the answer to a question, it queries 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`) is used.

If `port` is `*` or is omitted, a random port number from a pre-configured range is picked up and used for each query. The port range(s) is 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` checks whether the operating system provides a programming interface to retrieve the system's default range for ephemeral ports. If such an interface is available, `named` uses the corresponding system default range; otherwise, it uses its own defaults:

```
use-v4-udp-ports { range 1024 65535; };
use-v6-udp-ports { range 1024 65535; };
```

Note: Make sure the ranges are 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 is automatically applied when `named` is reloaded. Explicit configuration of use-v4-udp-ports and use-v6-udp-ports is encouraged, 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 do not allow `named`, if run 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: BIND 9.5.0 introduced the `use-queryport-pool` option to support a pool of such random ports, but this option is now obsolete because reusing the same ports in the pool may not be sufficiently secure. For the same reason, 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.

`use-queryport-pool` This option is obsolete.

`queryport-pool-ports` This option is obsolete.

`queryport-pool-updateinterval` This option is obsolete.

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: Solaris 2.5.1 and earlier does not support setting the source address for TCP sockets.

Note: See also `transfer-source` and `notify-source`.

Zone Transfers

BIND 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` This option defines a global list of IP addresses of name servers that are also sent 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 quickly converge on stealth servers. 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.

If an `also-notify` list is given in a `zone` statement, it overrides the `options also-notify` statement. When a `zone notify` statement is set to `no`, the IP addresses in the global `also-notify` list are not sent NOTIFY messages for that zone. The default is the empty list (no global notification list).

`max-transfer-time-in` Inbound zone transfers running longer than this many minutes are terminated. The default is 120 minutes (2 hours). The maximum value is 28 days (40320 minutes).

`max-transfer-idle-in` Inbound zone transfers making no progress in this many minutes are terminated. The default is 60 minutes (1 hour). The maximum value is 28 days (40320 minutes).

`max-transfer-time-out` Outbound zone transfers running longer than this many minutes are terminated. The default is 120 minutes (2 hours). The maximum value is 28 days (40320 minutes).

`max-transfer-idle-out` Outbound zone transfers making no progress in this many minutes are terminated. The default is 60 minutes (1 hour). The maximum value is 28 days (40320 minutes).

`notify-rate` This specifies the rate at which NOTIFY requests are sent during normal zone maintenance operations. (NOTIFY requests due to initial zone loading are subject to a separate rate limit; see below.) The default is 20 per second. The lowest possible rate is one per second; when set to zero, it is silently raised to one.

`startup-notify-rate` This is the rate at which NOTIFY requests are sent when the name server is first starting up, or when zones have been newly added to the name server. The default is 20 per second. The lowest possible rate is one per second; when set to zero, it is silently raised to one.

`serial-query-rate` Secondary servers periodically query primary servers to find out if zone serial numbers have changed. Each such query uses a minute amount of the secondary server's network bandwidth. To limit the amount of bandwidth used, BIND 9 limits the rate at which queries are sent. The value of the `serial-query-rate` option, an integer, is the maximum number of queries sent per second. The default is 20 per second. The lowest possible rate is one per second; when set to zero, it is silently raised to one.

`transfer-format` Zone transfers can be sent using two different formats, `one-answer` and `many-answers`. The `transfer-format` option is used on the primary server to determine which format it sends. `one-answer` uses one DNS message per resource record transferred. `many-answers` packs as many resource records as possible into a message. `many-answers` is more efficient; the default is `many-answers`. `transfer-format` may be overridden on a per-server basis by using the `server` statement.

`transfer-message-size` This is an upper bound on the uncompressed size of DNS messages used in zone transfers over TCP. If a message grows larger than this size, additional messages are used to complete the zone transfer. (Note, however, that this is a hint, not a hard limit; if a message contains a single resource record whose RDATA does not fit within the size limit, a larger message will be permitted so the record can be transferred.)

Valid values are between 512 and 65535 octets; any values outside that range are adjusted to the nearest value within it. The default is `20480`, which was selected to improve message compression; most DNS messages of this size will compress to less than 16536 bytes. Larger messages cannot be compressed as effectively, because 16536 is the largest permissible compression offset pointer in a DNS message.

This option is mainly intended for server testing; there is rarely any benefit in setting a value other than the default.

`transfers-in` This is the maximum number of inbound zone transfers that can be running concurrently. The default value is `10`. Increasing `transfers-in` may speed up the convergence of secondary zones, but it also may increase the load on the local system.

`transfers-out` This is the maximum number of outbound zone transfers that can be running concurrently. Zone transfer requests in excess of the limit are refused. The default value is `10`.

`transfers-per-ns` This is the maximum number of inbound zone transfers that can be concurrently transferring from a given remote name server. The default value is `2`. Increasing `transfers-per-ns` may speed up the convergence of secondary zones, but it also may increase the load on the remote name server. `transfers-per-ns` may be overridden on a per-server basis by using the `transfers` phrase of the `server` statement.

`transfer-source` `transfer-source` determines which local address is 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 refresh queries and forwarded dynamic updates. If not set, it defaults to a system controlled value which is usually the address of the interface "closest to" the remote end. This address must appear in the remote end's `allow-transfer` option for the zone being transferred, if one is specified. This statement sets the `transfer-source` for all zones, but can be overridden on a per-view or per-zone basis by including a `transfer-source` statement within the `view` or `zone` block in the configuration file.

Note: Solaris 2.5.1 and earlier does not support setting the source address for TCP sockets.

`transfer-source-v6` This option is the same as `transfer-source`, except zone transfers are performed using IPv6.

`alt-transfer-source` This indicates an alternate transfer source if the one listed in `transfer-source` fails and `use-alt-transfer-source` is set.

Note: To avoid using the alternate transfer source, set `use-alt-transfer-source` appropriately and do not depend upon getting an answer back to the first refresh query.

`alt-transfer-source-v6` This indicates an alternate transfer source if the one listed in `transfer-source-v6` fails and `use-alt-transfer-source` is set.

`use-alt-transfer-source` This indicates whether the alternate transfer sources should be used. If views are specified, this defaults to `no`; otherwise, it defaults to `yes`.

`notify-source` `notify-source` determines which local source address, and optionally UDP port, is used to send NOTIFY messages. This address must appear in the secondary server's `masters` zone clause or in an `allow-notify` clause. This statement sets the `notify-source` for all zones, but can be overridden on a per-zone or per-view basis by including a `notify-source` statement within the `zone` or `view` block in the configuration file.

Note: Solaris 2.5.1 and earlier does not support setting the source address for TCP sockets.

`notify-source-v6` This option acts like `notify-source`, but applies to notify messages sent to IPv6 addresses.

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 are or are 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` are in one of the following 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 a 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.

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 *Configuration File Elements*.

The following options set operating system resource limits for the name server process. Some operating systems do not support some or any of the limits; on such systems, a warning is issued if an unsupported limit is used.

`coresize` This sets the maximum size of a core dump. The default is `default`.

`datasize` This sets the maximum amount of data memory the server may use. The default 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 to limit 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. To limit the amount of memory used by the server, use the `max-cache-size` and `recursive-clients` options instead.

`files` This sets the maximum number of files the server may have open concurrently. The default is `unlimited`.

`stacksize` This sets the maximum amount of stack memory the server may use. The default is `default`.

Server Resource Limits

The following options set limits on the server's resource consumption that are enforced internally by the server rather than by the operating system.

`max-journal-size` This sets a maximum size for each journal file (see *The Journal File*), expressed in bytes or, if followed by an optional unit suffix ('k', 'm', or 'g'), in kilobytes, megabytes, or gigabytes. When the journal file approaches the specified size, some of the oldest transactions in the journal are automatically removed. The largest permitted value is 2 gigabytes. Very small values are rounded up to 4096 bytes. It is possible to specify `unlimited`, which also means 2 gigabytes. If the limit is set to

`default` or left unset, the journal is allowed to grow up to twice as large as the zone. (There is little benefit in storing larger journals.)

This option may also be set on a per-zone basis.

`max-records` This sets the maximum number of records permitted in a zone. The default is zero, which means the maximum is unlimited.

`recursive-clients` This sets the maximum number (a "hard quota") of simultaneous recursive lookups the server performs on behalf of clients. The default is `1000`. Because each recursing client uses a fair bit of memory (on the order of 20 kilobytes), the value of the `recursive-clients` option may have to be decreased on hosts with limited memory.

`recursive-clients` defines a "hard quota" limit for pending recursive clients; when more clients than this are pending, new incoming requests are not accepted, and for each incoming request a previous pending request is dropped.

A "soft quota" is also set. When this lower quota is exceeded, incoming requests are accepted, but for each one, a pending request is dropped. If `recursive-clients` is greater than 1000, the soft quota is set to `recursive-clients` minus 100; otherwise it is set to 90% of `recursive-clients`.

`tcp-clients` This is the maximum number of simultaneous client TCP connections that the server accepts. The default is `150`.

`clients-per-query; max-clients-per-query` These set the initial value (minimum) and maximum number of recursive simultaneous clients for any given query (<qname,qtype,qclass>) that the server accepts before dropping additional clients. `named` attempts to self-tune this value and changes are logged. The default values are 10 and 100.

This value should reflect how many queries come in for a given name in the time it takes to resolve that name. If the number of queries exceeds this value, `named` assumes that it is dealing with a non-responsive zone and drops additional queries. If it gets a response after dropping queries, it raises the estimate. The estimate is then lowered in 20 minutes if it has remained unchanged.

If `clients-per-query` is set to zero, there is no limit on the number of clients per query and no queries are dropped.

If `max-clients-per-query` is set to zero, there is no upper bound other than that imposed by `recursive-clients`.

`fetches-per-zone` This sets the maximum number of simultaneous iterative queries to any one domain that the server permits 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`.

When many clients simultaneously query for the same name and type, the clients are all attached to the same fetch, up to the `max-clients-per-query` limit, and only one iterative query is sent. However, when clients are simultaneously querying for *different* names or types, multiple queries are sent and `max-clients-per-query` is not effective as a limit.

Optionally, this value may be followed by the keyword `drop` or `fail`, indicating whether queries which exceed the fetch quota for a zone are dropped with no response, or answered with SERVFAIL. The default is `drop`.

If `fetches-per-zone` is set to zero, then there is no limit on the number of fetches per query and no queries are dropped. The default is zero.

The current list of active fetches can be dumped by running `rndc recursing`. 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.)

`fetches-per-server` This sets the maximum number of simultaneous iterative queries that the server allows to be sent to a single upstream name server before blocking additional queries. This value 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 smaller than `recursive-clients`.

Optionally, this value may be followed by the keyword `drop` or `fail`, indicating whether queries are dropped with no response, or answered with SERVFAIL, when all of the servers authoritative for a zone are found to have exceeded the per-server quota. The default is `fail`.

If `fetches-per-server` is set to zero, then there is no limit on the number of fetches per query and no queries are dropped. The default is zero.

The `fetches-per-server` quota is dynamically adjusted in response to detected congestion. As queries are sent to a server and either are answered or time out, 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 timeout ratio drops below a "low" threshold, then `fetches-per-server` is increased. The `fetch-quota-params` options can be used to adjust the parameters for this calculation.

`fetch-quota-params` This sets the parameters to use for dynamic resizing of the `fetches-per-server` quota in response to detected congestion.

The first argument is an integer value indicating how frequently to recalculate the moving average of the ratio of timeouts to responses for each server. The default is 100, meaning that BIND recalculates the average ratio after every 100 queries have either been answered or timed out.

The remaining three arguments represent the "low" threshold (defaulting to a timeout ratio of 0.1), the "high" threshold (defaulting to a timeout ratio of 0.3), and the discount rate for the moving average (defaulting to 0.7). 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. These arguments are all fixed-point numbers with precision of 1/100: at most two places after the decimal point are significant.

`reserved-sockets` This sets the number of file descriptors reserved for TCP, stdio, etc. This needs to be big enough to cover the number of interfaces `named` listens on plus `tcp-clients`, as well as to provide room for outgoing TCP queries and incoming zone transfers. The default is `512`. The minimum value is `128` and the maximum value is `128` fewer than maxsockets (-S). This option may be removed in the future.

This option has little effect on Windows.

`max-cache-size` This sets the maximum amount of memory to use for the server's cache, in bytes or percentage of total physical memory. When the amount of data in the cache reaches this limit, the server causes records to expire prematurely based on an LRU-based strategy so that the limit is not exceeded. The keyword `unlimited`, or the value 0, places no limit on the cache size; records are purged from the cache only when their TTLs expire. Any positive values less than 2MB are ignored and reset to 2MB. In a server with multiple views, the limit applies separately to the cache of each view. The default is `90%`. On systems where detection of the amount of physical memory is not supported, values represented as a percentage fall back to unlimited. Note that the detection of physical memory is done only once at startup, so `named` does not adjust the cache size if the amount of physical memory is changed during runtime.

`tcp-listen-queue` This sets the listen queue depth. The default and minimum is 10. If the kernel supports the accept filter "dataready", this also controls how many TCP connections that are queued in kernel space waiting for some data before being passed to accept. Nonzero values less than 10 are silently

raised. A value of 0 may also be used; on most platforms this sets the listen queue length to a system-defined default value.

`tcp-initial-timeout` This sets the amount of time (in units of 100 milliseconds) that the server waits on a new TCP connection for the first message from the client. The default is 300 (30 seconds), the minimum is 25 (2.5 seconds), and the maximum is 1200 (two minutes). Values above the maximum or below the minimum are adjusted with a logged warning. (Note: This value must be greater than the expected round-trip delay time; otherwise, no client will ever have enough time to submit a message.) This value can be updated at runtime by using `rndc tcp-timeouts`.

`tcp-idle-timeout` This sets the amount of time (in units of 100 milliseconds) that the server waits on an idle TCP connection before closing it, when the client is not using the EDNS TCP keepalive option. The default is 300 (30 seconds), the maximum is 1200 (two minutes), and the minimum is 1 (one-tenth of a second). Values above the maximum or below the minimum are adjusted with a logged warning. See `tcp-keepalive-timeout` for clients using the EDNS TCP keepalive option. This value can be updated at runtime by using `rndc tcp-timeouts`.

`tcp-keepalive-timeout` This sets the amount of time (in units of 100 milliseconds) that the server waits on an idle TCP connection before closing it, when the client is using the EDNS TCP keepalive option. The default is 300 (30 seconds), the maximum is 65535 (about 1.8 hours), and the minimum is 1 (one-tenth of a second). Values above the maximum or below the minimum are adjusted with a logged warning. This value may be greater than `tcp-idle-timeout` because clients using the EDNS TCP keepalive option are expected to use TCP connections for more than one message. This value can be updated at runtime by using `rndc tcp-timeouts`.

`tcp-advertised-timeout` This sets the timeout value (in units of 100 milliseconds) that the server sends in responses containing the EDNS TCP keepalive option, which informs a client of the amount of time it may keep the session open. The default is 300 (30 seconds), the maximum is 65535 (about 1.8 hours), and the minimum is 0, which signals that the clients must close TCP connections immediately. Ordinarily this should be set to the same value as `tcp-keepalive-timeout`. This value can be updated at runtime by using `rndc tcp-timeouts`.

Periodic Task Intervals

`cleaning-interval` This option is obsolete.

`heartbeat-interval` The server performs zone maintenance tasks for all zones marked as `dialup` whenever this interval expires. The default 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 occurs.

`interface-interval` The server scans the network interface list every `interface-interval` minutes. The default is 60 minutes; the maximum value is 28 days (40320 minutes). If set to 0, interface scanning only occurs when the configuration file is loaded, or when `automatic-interface-scan` is enabled and supported by the operating system. After the scan, the server begins listening for queries on any newly discovered interfaces (provided they are allowed by the `listen-on` configuration), and stops listening on interfaces that have gone away. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

The `sortlist` Statement

The response to a DNS query may consist of multiple resource records (RRs) forming a resource record set (RRset). The name server normally returna the RRs within the RRset in an indeterminate order (but see the `rrset-order` statement in *RRset Ordering*). The client resolver code should rearrange the RRs as appropriate; that is, using any addresses on the local net in preference to other addresses. However, not all resolvers can do this or are correctly configured. When a client is using a local server, the sorting can be

performed in the server, based on the client's address. This only requires configuring the name servers, not all the clients.

The `sortlist` statement (see below) takes an `address_match_list` and interprets it in a special way. Each top-level statement in the `sortlist` must itself be an explicit `address_match_list` with one or two elements. The first element (which may be an IP address, an IP prefix, an ACL name, or a nested `address_match_list`) of each top-level list is checked against the source address of the query until a match is found. When the addresses in the first element overlap, the first rule to match is selected.

Once the source address of the query has been matched, if the top level statement contains only one element, the actual primitive element that matched the source address is used to select the address in the response to move to the beginning of the response. If the statement is a list of two elements, then the second element is interpreted as a topology preference list. Each top-level element is assigned a distance, and the address in the response with the minimum distance is moved to the beginning of the response.

In the following example, any queries received from any of the addresses of the host itself will get responses preferring addresses on any of the locally connected networks. Next most preferred are addresses on the 192.168.1/24 network, and after that either the 192.168.2/24 or 192.168.3/24 network with no preference shown between these two networks. Queries received from a host on the 192.168.1/24 network will prefer other addresses on that network to the 192.168.2/24 and 192.168.3/24 networks. Queries received from a host on the 192.168.4/24 or the 192.168.5/24 network will only prefer other addresses on their directly connected networks.

```
sortlist {
    // IF the local host
    // THEN first fit on the following nets
    { localhost;
    { localnets;
        192.168.1/24;
        { 192.168.2/24; 192.168.3/24; }; }; };
    // IF on class C 192.168.1 THEN use .1, or .2 or .3
    { 192.168.1/24;
    { 192.168.1/24;
        { 192.168.2/24; 192.168.3/24; }; }; };
    // IF on class C 192.168.2 THEN use .2, or .1 or .3
    { 192.168.2/24;
    { 192.168.2/24;
        { 192.168.1/24; 192.168.3/24; }; }; };
    // IF on class C 192.168.3 THEN use .3, or .1 or .2
    { 192.168.3/24;
    { 192.168.3/24;
        { 192.168.1/24; 192.168.2/24; }; }; };
    // IF .4 or .5 THEN prefer that net
    { { 192.168.4/24; 192.168.5/24; };
    };
};
```

The following example illlustrates reasonable behavior for the local host and hosts on directly connected networks. Responses sent to queries from the local host favor any of the directly connected networks. Responses sent to queries from any other hosts on a directly connected network prefer addresses on that same network. Responses to other queries are not sorted.

```
sortlist {
       { localhost; localnets; };
       { localnets; };
```

(continues on next page)

(continued from previous page)

```
};
```

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. The `rrset-order` statement permits configuration of the ordering of the records in a multiple-record response. See also the `sortlist` statement, *The sortlist Statement*.

An `order_spec` is defined as follows:

[class *class_name*] [type *type_name*] [name "*domain_name*"] order *ordering*

If no class is specified, the default is `ANY`. If no type is specified, the default is `ANY`. If no name is specified, the default is "`*`" (asterisk).

The legal values for `ordering` are:

`fixed` Records are returned in the order they are defined in the zone file. This option is only available if BIND is configured with "–enable-fixed-rrset" at compile time.

`random` Records are returned in a random order.

`cyclic` Records are returned in a cyclic round-robin order, rotating by one record per query. If BIND is configured with "–enable-fixed-rrset" at compile time, the initial ordering of the RRset matches the one specified in the zone file; otherwise the initial ordering is indeterminate.

`none` Records are returned in whatever order they were retrieved from the database. This order is indeterminate, but remains consistent as long as the database is not modified. When no ordering is specified, this is the default.

For example:

```
rrset-order {
   class IN type A name "host.example.com" order random;
   order cyclic;
};
```

causes any responses for type A records in class IN, that have `host.example.com` as a suffix, to always be returned in random order. All other records are returned in cyclic order.

If multiple `rrset-order` statements appear, they are not combined; the last one applies.

By default, records are returned in `random` order.

Note: "Fixed" ordering of the `rrset-order` statement by default is not currently supported in BIND 9. Fixed ordering can be enabled at compile time by specifying "–enable-fixed-rrset" on the "configure" command line.

Tuning

`lame-ttl` This sets the number of seconds to cache a lame server indication. 0 disables caching. (This is **NOT** recommended.) The default is `600` (10 minutes) and the maximum value is `1800` (30 minutes).

`servfail-ttl` This sets the number of seconds to cache a SERVFAIL response due to DNSSEC validation failure or other general server failure. If set to `0`, SERVFAIL caching is disabled. The SERVFAIL cache

is not consulted if a query has the CD (Checking Disabled) bit set; this allows a query that failed due to DNSSEC validation to be retried without waiting for the SERVFAIL TTL to expire.

The maximum value is `30` seconds; any higher value is silently reduced. The default is `1` second.

`min-ncache-ttl` To reduce network traffic and increase performance, the server stores negative answers. `min-ncache-ttl` is used to set a minimum retention time for these answers in the server in seconds. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

The default `min-ncache-ttl` is `0` seconds. `min-ncache-ttl` cannot exceed 90 seconds and is truncated to 90 seconds if set to a greater value.

`min-cache-ttl` This sets the minimum time for which the server caches ordinary (positive) answers, in seconds. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

The default `min-cache-ttl` is `0` seconds. `min-cache-ttl` cannot exceed 90 seconds and is truncated to 90 seconds if set to a greater value.

`max-ncache-ttl` To reduce network traffic and increase performance, the server stores negative answers. `max-ncache-ttl` is used to set a maximum retention time for these answers in the server in seconds. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

The default `max-ncache-ttl` is 10800 seconds (3 hours). `max-ncache-ttl` cannot exceed 7 days and is silently truncated to 7 days if set to a greater value.

`max-cache-ttl` This sets the maximum time for which the server caches ordinary (positive) answers, in seconds. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

The default `max-cache-ttl` is 604800 (one week). A value of zero 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.

`max-stale-ttl` If retaining stale RRsets in cache is enabled, and returning of stale cached answers is also enabled, `max-stale-ttl` sets the maximum time for which the server retains records past their normal expiry to return them as stale records, when the servers for those records are not reachable. The default is 12 hours. The minimum allowed is 1 second; a value of 0 is updated silently to 1 second.

For stale answers to be returned, the retaining of them in cache must be enabled via the configuration option `stale-cache-enable`, and returning cached answers must be enabled, either in the configuration file using the `stale-answer-enable` option or by calling `rndc serve-stale on`.

When `stale-cache-enable` is set to `no`, setting the `max-stale-ttl` has no effect, the value of `max-cache-ttl` will be `0` in such case.

`resolver-nonbackoff-tries` This specifies how many retries occur before exponential backoff kicks in. The default is `3`.

`resolver-retry-interval` This sets the base retry interval in milliseconds. The default is `800`.

`sig-validity-interval` This specifies the number of days into the future when DNSSEC signatures automatically generated as a result of dynamic updates (*Dynamic Update*) will expire. There is an optional second field which specifies how long before expiry the signatures are regenerated. If not specified, the signatures are regenerated at 1/4 of the base interval. The second field is specified in days if the base interval is greater than 7 days; otherwise it is specified in hours. The default base interval is `30` days, giving a re-signing interval of 7 1/2 days. The maximum value is 10 years (3660 days).

The signature inception time is unconditionally set to one hour before the current time, to allow for a limited amount of clock skew.

The `sig-validity-interval` can be overridden for DNSKEY records by setting `dnskey-sig-validity`.

The `sig-validity-interval` should be at least several multiples of the SOA expire interval, to allow for reasonable interaction between the various timer and expiry dates.

`dnskey-sig-validity` This specifies the number of days into the future when DNSSEC signatures that are automatically generated for DNSKEY RRsets as a result of dynamic updates (*Dynamic Update*) will expire. If set to a non-zero value, this overrides the value set by `sig-validity-interval`. The default is zero, meaning `sig-validity-interval` is used. The maximum value is 3660 days (10 years), and higher values are rejected.

`sig-signing-nodes` This specifies the maximum number of nodes to be examined in each quantum when signing a zone with a new DNSKEY. The default is `100`.

`sig-signing-signatures` This specifies a threshold number of signatures that terminates processing a quantum when signing a zone with a new DNSKEY. The default is `10`.

`sig-signing-type` This specifies a private RDATA type to be used when generating signing state records. The default is `65534`.

It is expected that this parameter may be removed in a future version once there is a standard type.

Signing state records are used to internally 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 command `rndc signing -list zone`. 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 by running `rndc signing -clear keyid/algorithm zone`. To clear all of the completed signing state records for a zone, use `rndc signing -clear all zone`.

`min-refresh-time; max-refresh-time; min-retry-time; max-retry-time` These options control 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 primary, giving secondary server administrators little control over their contents.

These options allow the administrator to set a minimum and maximum refresh and retry time in seconds per-zone, per-view, or globally. These options are valid for secondary and stub zones, and clamp the SOA refresh and retry times to the specified values.

The following defaults apply: `min-refresh-time` 300 seconds, `max-refresh-time` 2419200 seconds (4 weeks), `min-retry-time` 500 seconds, and `max-retry-time` 1209600 seconds (2 weeks).

`edns-udp-size` This sets the maximum advertised EDNS UDP buffer size in bytes, to control the size of packets received from authoritative servers in response to recursive queries. Valid values are 512 to 4096; values outside this range are silently adjusted to the nearest value within it. The default value is 4096.

The usual reason for setting `edns-udp-size` to a non-default value is to get 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 advertises a UDP buffer size of 512, as this has the greatest chance of success on the first try.

If the initial query is successful with EDNS advertising a buffer size of 512, then `named` will advertise progressively larger buffer sizes on successive queries, until responses begin timing out or `edns-udp-size` is reached.

The default buffer sizes used by `named` are 512, 1232, 1432, and 4096, but never exceeding `edns-udp-size`. (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.)

max-udp-size This sets the maximum EDNS UDP message size that named sends in bytes. Valid values are 512 to 4096; values outside this range are silently adjusted to the nearest value within it. The default value is 4096.

This value applies to responses sent by a server; to set the advertised buffer size in queries, see edns-udp-size.

The usual reason for setting max-udp-size 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 is independent of the advertised receive buffer (edns-udp-size).

Setting this to a low value encourages additional TCP traffic to the name server.

masterfile-format This specifies the file format of zone files (see *Additional File Formats*). The default value is text, which is the standard textual representation, except for secondary zones, in which the default value is raw. Files in formats other than text are typically expected to be generated by the named-compilezone tool, or dumped by named.

Note that when a zone file in a format other than text is loaded, named may omit some of the checks which are performed for a file in the text format. In particular, check-names checks do not apply for the raw format. This means a zone file in the raw format must be generated with the same check level as that specified in the named configuration file. Also, map format files are loaded directly into memory via memory mapping, with only minimal checking.

This statement sets the masterfile-format for all zones, but can be overridden on a per-zone or per-view basis by including a masterfile-format statement within the zone or view block in the configuration file.

masterfile-style This specifies the formatting of zone files during dump when the masterfile-format is text. This option is ignored with any other masterfile-format.

When set to relative, records are printed in a multi-line format, with owner names expressed relative to a shared origin. When set to full, records are printed in a single-line format with absolute owner names. The full format is most suitable when a zone file needs to be processed automatically by a script. The relative format is more human-readable, and is thus suitable when a zone is to be edited by hand. The default is relative.

max-recursion-depth This 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 a name server address, which in turn requires resolving another name, etc.; if the number of recursion exceeds this value, the recursive query is terminated and returns SERVFAIL. The default is 7.

max-recursion-queries This 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. The default is 75.

notify-delay This sets the delay, in seconds, between sending sets of notify messages for a zone. The default is 5 seconds.

The overall rate that NOTIFY messages are sent for all zones is controlled by serial-query-rate.

max-rsa-exponent-size This sets the maximum RSA exponent size, in bits, that is accepted when validating. Valid values are 35 to 4096 bits. The default, zero, is also accepted and is equivalent to 4096.

prefetch 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 always has an answer available.

The prefetch specifies the "trigger" TTL value at which prefetch of the current query takes place; when a cache record with a lower TTL value is encountered during query processing, it is refreshed.

Valid trigger TTL values are 1 to 10 seconds. Values larger than 10 seconds are silently reduced to 10. Setting a trigger TTL to zero causes prefetch to be disabled. The default trigger TTL is `2`.

An optional second argument specifies the "eligibility" TTL: the smallest *original* TTL value that is accepted for a record to be eligible for prefetching. The eligibility TTL must be at least six seconds longer than the trigger TTL; if not, `named` silently adjusts it upward. The default eligibility TTL is `9`.

`v6-bias` When determining the next name server to try, this indicates by how many milliseconds to prefer IPv6 name servers. The default is `50` milliseconds.

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 `bind` in the `CHAOS` class. These zones are part of a built-in view (see *view Statement Grammar*) of class `CHAOS` which is separate from the default view of class `IN`. Most global configuration options (`allow-query`, etc.) 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.

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` This is the version the server should report via a query of the name `version.bind` with type `TXT` and class `CHAOS`. The default is the real version number of this server. Specifying `version none` disables processing of the queries.

Setting `version` to any value (including `none`) also disables queries for `authors.bind TXT CH`.

`hostname` This is the hostname the server should report via a query of the name `hostname.bind` with type `TXT` and class `CHAOS`. This defaults to the hostname of the machine hosting the name server, as found by the gethostname() function. The primary purpose of such queries is to identify which of a group of anycast servers is actually answering the queries. Specifying `hostname none;` disables processing of the queries.

`server-id` This is the ID the server should report when receiving a Name Server Identifier (NSID) query, or a query of the name `ID.SERVER` with type `TXT` and class `CHAOS`. The primary purpose of such queries is to identify which of a group of anycast servers is actually answering the queries. Specifying `server-id none;` disables processing of the queries. Specifying `server-id hostname;` causes `named` to use the hostname as found by the gethostname() function. The default `server-id` is `none`.

Built-in Empty Zones

The `named` server has some built-in empty zones, for SOA and NS records only. These are for zones that should normally be answered locally and for which queries should not be sent to the Internet's root servers. The official servers that cover these namespaces return NXDOMAIN responses to these queries. In particular, these cover the reverse namespaces for addresses from **RFC 1918**, **RFC 4193**, **RFC 5737**, and **RFC 6598**. They also include the reverse namespace for the IPv6 local address (locally assigned), IPv6 link local addresses, the IPv6 loopback address, and the IPv6 unknown address.

The server attempts to determine if a built-in zone already exists or is active (covered by a forward-only forwarding declaration) and does 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
- 123.100.IN-ADDR.ARPA
- 124.100.IN-ADDR.ARPA
- 125.100.IN-ADDR.ARPA
- 126.100.IN-ADDR.ARPA
- 127.100.IN-ADDR.ARPA
- 0.IN-ADDR.ARPA
- 127.IN-ADDR.ARPA
- 254.169.IN-ADDR.ARPA
- 2.0.192.IN-ADDR.ARPA
- 100.51.198.IN-ADDR.ARPA
- 113.0.203.IN-ADDR.ARPA
- 255.255.255.255.IN-ADDR.ARPA
- 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA
- 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA
- 8.B.D.0.1.0.0.2.IP6.ARPA
- D.F.IP6.ARPA
- 8.E.F.IP6.ARPA
- 9.E.F.IP6.ARPA
- A.E.F.IP6.ARPA
- B.E.F.IP6.ARPA
- EMPTY.AS112.ARPA
- HOME.ARPA

Empty zones can be set at the view level and only apply to views of class IN. Disabled empty zones are only inherited from options if there are no disabled empty zones specified at the view level. To override the options list of disabled zones, disable the root zone at the view level. For example:

```
disable-empty-zone ".";
```

If using the address ranges covered here, reverse zones covering the addresses should already be in place. 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 had 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 zones under the parent zone they serve. For the real root servers, this is all built-in empty zones. This enables them to return referrals to deeper in the tree.

---

empty-server This specifies the server name that appears in the returned SOA record for empty zones. If none is specified, the zone's name is used.

empty-contact This specifies the contact name that appears in the returned SOA record for empty zones. If none is specified, "." is used.

empty-zones-enable This enables or disables all empty zones. By default, they are enabled.

disable-empty-zone This disables individual empty zones. By default, none are disabled. This option can be specified multiple times.

Content Filtering

BIND 9 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 are accepted regardless of the filter setting. Likewise, if the alias name is a subdomain of the corresponding zone, the `deny-answer-aliases` filter does 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` is silently ignored.

If a response message is rejected due to the filtering, the entire message is discarded without being cached, and a SERVFAIL error is 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 the user's own network or an alias name within the user's 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 the local network that could not be externally accessed otherwise. See the paper available at https://dl.acm.org/doi/10.1145/1315245.1315298 for more details about the attacks.

For example, with a domain named "example.net" and an internal network using an IPv4 prefix 192.0.2.0/24, an administrator 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 let a web browser in the 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 would be ignored.

On the other hand, if the browser looked up a legitimate internal web server "www.example.net" and the following response were returned to the BIND 9 server:

```
www.example.net. A 192.0.2.2
```

it would 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 with having an "external" name mapped to an "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 either is not possible or does not make sense to detect whether the intent of the mapping is legitimate 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 there is no other choice and the attack is a real threat to applications.

Care should be particularly taken if using 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.

Response Policy Zone (RPZ) Rewriting

BIND 9 includes a limited mechanism to modify DNS responses for requests analogous to email anti-spam DNS rejection lists. 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; };`. Note that zones using `masterfile-format map` cannot be used as policy zones.

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 64 on the number of policy zones in a single `response-policy` option; more than that is a configuration error.

Rules encoded in response policy zones are processed after those defined in *Access Control*. All queries from clients which are not permitted access to the resolver are answered with a status code of REFUSED, regardless of configured RPZ rules.

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 that 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 addresses 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. The `nsdname-enable` phrase turns NSDNAME triggers off or on for a single policy zone or for all zones.

If authoritative name servers for the query name are not yet known, `named` recursively looks up the authoritative servers for the query name before applying an RPZ-NSDNAME rule. This can cause a processing delay. To speed up processing at the cost of precision, the `nsdname-wait-recurse` option can be used; when set to `no`, RPZ-NSDNAME rules are only applied when authoritative servers for the query name have already been looked up and cached. If authoritative servers for the query name are not in the cache, the RPZ-NSDNAME rule is ignored, but the authoritative servers for the query name are looked up in the background and the rule is applied to subsequent queries. The default is `yes`, meaning RPZ-NSDNAME rules are always applied even if authoritative servers for the query name need to be looked up first.

`RPZ-NSIP` NSIP triggers match the IP addresses of authoritative servers. They 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 `nsip-enable` phrase turns NSIP triggers off or on for a single policy zone or for all zones.

If a name server's IP address is not yet known, `named` recursively looks up the IP address before applying an RPZ-NSIP rule. This can cause a processing delay. To speed up processing at the cost of precision, the `nsip-wait-recurse` option can be used; when set to `no`, RPZ-NSIP rules are only applied when a name server's IP address has already been looked up and cached. If a server's IP address is not in the cache, the RPZ-NSIP rule is ignored, but the address is looked up in the background and the rule is applied to subsequent queries. The default is `yes`, meaning RPZ-NSIP rules are always applied even if an address needs to be looked up first.

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 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 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 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 a CNAME whose target is the wildcard top-level domain (`*.`). It rewrites the response to NODATA or ANCOUNT=0.

`Local Data` A set of ordinary DNS records can be used to answer queries. Queries for record types not in 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 an ordinary CNAME after the asterisk (*) has been replaced with the query name. This special form is useful for query logging in the walled garden's authoritative 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 is written (or not) according to any triggered policy records that are not disabled. Disabled policy zones should appear first, because they are often not logged if a higher-precedence trigger is found first.

`PASSTHRU; DROP; TCP-Only; NXDOMAIN; NODATA` These settings each override the corresponding per-record policy.

`CNAME domain` This 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 for 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 the requested 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 behavior can cost significant time. The `qname-wait-recurse yes` option overrides the default and enables that 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 NSDNAME 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 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 `dnsrps-enable yes` option turns on the DNS Response Policy Service (DNSRPS) interface, if it has been compiled in `named` using `configure --enable-dnsrps`.

The `dnsrps-options` block provides additional RPZ configuration settings, which are passed through to the DNSRPS provider library. Multiple DNSRPS settings in an `dnsrps-options` string should be separated with semi-colons. The DNSRPS provider, librpz, is passed a configuration string consisting of the `dnsrps-options` text, concatenated with settings derived from the `response-policy` statement.

Note: The `dnsrps-options` text should only include configuration settings that are specific to the DNSRPS provider. For example, the DNSRPS provider from Farsight Security takes options such as `dnsrpzd-conf`, `dnsrpzd-sock`, and `dnzrpzd-args` (for details of these options, see the `librpz` documentation). Other RPZ configuration settings could be included in `dnsrps-options` as well, but if `named` were switched back to traditional RPZ by setting `dnsrps-enable` to "no", those options would be ignored.

The TTL of a record modified by RPZ policies is set from the TTL of the relevant record in the policy zone. It is then limited to a maximum value. The `max-policy-ttl` clause changes the maximum seconds from its default of 5. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

For example, an administrator 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
;       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   .

; disapprove and approve some DNS clients
```

(continues on next page)

(continued from previous page)

```
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 BIND 9 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.

The `log` clause can be used to optionally turn off rewrite logging for a particular response policy zone. By default, all rewrites are logged.

The `add-soa` option controls whether the RPZ's SOA record is added to the section for traceback of changes from this zone. This can be set at the individual policy zone level or at the response-policy level. The default is `yes`.

Updates to RPZ zones are processed asynchronously; if there is more than one update pending they are bundled together. If an update to a RPZ zone (for example, via IXFR) happens less than `min-update-interval` seconds after the most recent update, the changes are not carried out until this interval has elapsed. The default is `60` seconds. For convenience, TTL-style time unit suffixes may be used to specify the value. It also accepts ISO 8601 duration formats.

Response Rate Limiting

Excessive almost-identical UDP *responses* can be controlled by configuring a `rate-limit` clause in an `options` or `view` statement. This mechanism keeps authoritative BIND 9 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 `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 rejection lists. 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 BIND 9 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, rather, all responses are dropped. A value of 1 causes every response to slip; values between 2 and 10 cause every nth 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, 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 supplements, 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 <IMG> 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 are truncated by rate limits are included in `RateSlipped` and `RespTruncated`.

`named` supports NXDOMAIN redirection via two methods:

- Redirect zone (*zone Statement Grammar*)
- Redirect namespace

With either method, when `named` gets a NXDOMAIN response it examines a separate namespace to see if the NXDOMAIN response should be replaced with an alternative response.

With a redirect zone (`zone "." { type redirect; };`), the data used to replace the NXDOMAIN is held in a single zone which is not part of the normal namespace. All the redirect information is contained in the zone; there are no delegations.

With a redirect namespace (`option { nxdomain-redirect <suffix> };`) the data used to replace the NXDOMAIN is part of the normal namespace and is looked up by appending the specified suffix to the original query name. This roughly doubles the cache required to process NXDOMAIN responses, as both the original NXDOMAIN response and the replacement data (or an NXDOMAIN indicating that there is no replacement) must be stored.

If both a redirect zone and a redirect namespace are configured, the redirect zone is tried first.

4.2.15 `server` Statement Grammar

```
server <netprefix> {
      bogus <boolean>;
      edns <boolean>;
      edns-udp-size <integer>;
      edns-version <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> ];
      padding <integer>;
      provide-ixfr <boolean>;
      query-source ( ( [ address ] ( <ipv4_address> | * ) [ port (
          <integer> | * ) ] ) | ( [ [ address ] ( <ipv4_address> | * ) ]
          port ( <integer> | * ) ) ) [ dscp <integer> ];
```

(continues on next page)

(continued from previous page)

```
    query-source-v6 ( ( [ address ] ( <ipv6_address> | * ) [ port (
        <integer> | * ) ] ) | ( [ [ address ] ( <ipv6_address> | * ) ]
        port ( <integer> | * ) ) ) [ dscp <integer> ];
    request-expire <boolean>;
    request-ixfr <boolean>;
    request-nsid <boolean>;
    send-cookie <boolean>;
    tcp-keepalive <boolean>;
    tcp-only <boolean>;
    transfer-format ( many-answers | one-answer );
    transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [
        dscp <integer> ];
    transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * )
        ] [ dscp <integer> ];
    transfers <integer>;
};
```

4.2.16 `server` Statement Definition and Usage

The `server` statement defines characteristics to be associated with a remote name server. If a prefix length is specified, then a range of servers is covered. Only the most specific server clause applies, regardless of the order in `named.conf`.

The `server` statement can occur at the top level of the configuration file or inside 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 as defaults.

If a remote server is giving out bad data, marking it as bogus prevents further queries to it. The default value of `bogus` is `no`.

The `provide-ixfr` clause determines whether the local server, acting as master, responds with an incremental zone transfer when the given remote server, a secondary, requests it. If set to `yes`, incremental transfer is provided whenever possible. If set to `no`, all transfers to the remote server are non-incremental. If not set, the value of the `provide-ixfr` option in the view or global options block is used as a default.

The `request-ixfr` clause determines whether the local server, acting as a secondary, requests incremental zone transfers from the given remote server, a primary. If not set, the value of the `request-ixfr` option in the view or global options block is used as a default. It may also be set in the zone block; if set there, it overrides the global or view setting for that zone.

IXFR requests to servers that do not support IXFR automatically fall back to AXFR. Therefore, there is no need to manually list which servers support IXFR and which ones do not; the global 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 primary and secondary claim to support it: for example, if one of the servers is buggy and crashes or corrupts data when IXFR is used.

The `request-expire` clause determines whether the local server, when acting as a secondary, requests the EDNS EXPIRE value. The EDNS EXPIRE value indicates the remaining time before the zone data expires and needs to be refreshed. This is used when a secondary server transfers a zone from another secondary server; when transferring from the primary, the expiration timer is set from the EXPIRE field of the SOA record instead. The default is `yes`.

The `edns` clause determines whether the local server attempts to use EDNS when communicating with the remote server. The default is `yes`.

The `edns-udp-size` option sets the EDNS UDP size that is advertised by `named` when querying the remote server. Valid values are 512 to 4096 bytes; values outside this range are silently adjusted to the nearest value within it. This option is useful when advertising a different value to this server than the value advertised globally: for example, when there is a firewall at the remote site that is blocking large replies. (Note: Currently, this sets a single UDP size for all packets sent to the server; `named` does not deviate from this value. This differs from the behavior of `edns-udp-size` in `options` or `view` statements, where it specifies a maximum value. The `server` statement behavior may be brought into conformance with the `options`/`view` behavior in future releases.)

The `edns-version` option sets the maximum EDNS VERSION that is sent to the server(s) by the resolver. The actual EDNS version sent is still subject to normal EDNS version negotiation rules (see **RFC 6891**), the maximum EDNS version supported by the server, and any other heuristics that indicate that a lower version should be sent. This option is intended to be used when a remote server reacts badly to a given EDNS version or higher; it should be set to the highest version the remote server is known to support. Valid values are 0 to 255; higher values are silently adjusted. This option is not needed until higher EDNS versions than 0 are in use.

The `max-udp-size` option sets the maximum EDNS UDP message size `named` sends. Valid values are 512 to 4096 bytes; values outside this range are silently adjusted. This option is useful when there is a firewall that is blocking large replies from `named`.

The `padding` option adds EDNS Padding options to outgoing messages, increasing the packet size to a multiple of the specified block size. Valid block sizes range from 0 (the default, which disables the use of EDNS Padding) to 512 bytes. Larger values are reduced to 512, with a logged warning. Note: This option is not currently compatible with no TSIG or SIG(0), as the EDNS OPT record containing the padding would have to be added to the packet after it had already been signed.

The `tcp-only` option sets the transport protocol to TCP. The default is to use the UDP transport and to fallback on TCP only when a truncated response is received.

The `tcp-keepalive` option adds EDNS TCP keepalive to messages sent over TCP. Note that currently idle timeouts in responses are ignored.

The server supports two zone transfer methods. The first, `one-answer`, uses one DNS message per resource record transferred. `many-answers` packs as many resource records as possible into a message, which is more efficient. It is possible to specify which method to use for a server via the `transfer-format` option; if not set there, the `transfer-format` specified by the `options` statement is used.

`transfers` is used to limit the number of concurrent inbound zone transfers from the specified server. If no `transfers` clause is specified, the limit is set according to the `transfers-per-ns` option.

The `keys` clause identifies a `key_id` defined by the `key` statement, to be used for transaction security (see *TSIG*) when talking to the remote server. When a request is sent to the remote server, a request signature is generated using the key specified here and appended to the message. A request originating from the remote server is not required to be signed by this key.

Only a single key per server is currently supported.

The `transfer-source` and `transfer-source-v6` clauses specify the IPv4 and IPv6 source address to be used for zone transfer with the remote server, respectively. For an IPv4 remote server, only `transfer-source` can be specified. Similarly, for an IPv6 remote server, only `transfer-source-v6` can be specified. For more details, see the description of `transfer-source` and `transfer-source-v6` in *Zone Transfers*.

The `notify-source` and `notify-source-v6` clauses specify the IPv4 and IPv6 source address to be used for notify messages sent to remote servers, respectively. For an IPv4 remote server, only `notify-source` can be specified. Similarly, for an IPv6 remote server, only `notify-source-v6` can be specified.

The `query-source` and `query-source-v6` clauses specify the IPv4 and IPv6 source address to be used for queries sent to remote servers, respectively. For an IPv4 remote server, only `query-source` can be specified. Similarly, for an IPv6 remote server, only `query-source-v6` can be specified.

The `request-nsid` clause determines whether the local server adds a NSID EDNS option to requests sent to the server. This overrides `request-nsid` set at the view or option level.

The `send-cookie` clause determines whether the local server adds a COOKIE EDNS option to requests sent to the server. This overrides `send-cookie` set at the view or option level. The `named` server may determine that COOKIE is not supported by the remote server and not add a COOKIE EDNS option to requests.

4.2.17 `statistics-channels` Statement Grammar

```
statistics-channels {
      inet ( <ipv4_address> | <ipv6_address> |
          * ) [ port ( <integer> | * ) ] [
          allow { <address_match_element>; ...
          } ];
};
```

4.2.18 `statistics-channels` Statement Definition and Usage

The `statistics-channels` statement declares communication channels to be used by system administrators to get access to statistics information of the name server.

This statement is intended to be flexible to support multiple communication protocols in the future, but currently only HTTP access is supported. It requires that BIND 9 be compiled with libxml2 and/or json-c (also known as libjson0); the `statistics-channels` statement is still accepted even if it is built without the library, but any HTTP access fails with an error.

An `inet` control channel is a TCP socket listening at the specified `ip_port` on the specified `ip_addr`, which can be an IPv4 or IPv6 address. An `ip_addr` of `*` (asterisk) is interpreted as the IPv4 wildcard address; connections are accepted on any of the system's IPv4 addresses. To listen on the IPv6 wildcard address, use an `ip_addr` of `::`.

If no port is specified, port 80 is used for HTTP channels. The asterisk (`*`) cannot be used for `ip_port`.

Attempts to open a statistics channel are restricted by the optional `allow` clause. Connections to the statistics channel are permitted based on the `address_match_list`. If no `allow` clause is present, `named` accepts connection attempts from any address; since the statistics may contain sensitive internal information, it is highly recommended to restrict the source of connection requests appropriately.

If no `statistics-channels` statement is present, `named` does not open any communication channels.

The statistics are available in various formats and views depending on the URI used to access them. For example, if the statistics channel is configured to listen on 127.0.0.1 port 8888, then the statistics are accessible in XML format at http://127.0.0.1:8888/ or http://127.0.0.1:8888/xml. A CSS file is included, which can format the XML statistics into tables when viewed with a stylesheet-capable browser, and into charts and graphs using the Google Charts API when using a JavaScript-capable browser.

Broken-out subsets of the statistics can be viewed at http://127.0.0.1:8888/xml/v3/status (server uptime and last reconfiguration time), http://127.0.0.1:8888/xml/v3/server (server and resolver statistics), http://127.0.0.1:8888/xml/v3/zones (zone statistics), http://127.0.0.1:8888/xml/v3/net (network status and socket statistics), http://127.0.0.1:8888/xml/v3/mem (memory manager statistics), http://127.0.0.1:8888/xml/v3/tasks (task manager statistics), and http://127.0.0.1:8888/xml/v3/traffic (traffic sizes).

The full set of statistics can also be read in JSON format at http://127.0.0.1:8888/json, with the broken-out subsets at http://127.0.0.1:8888/json/v1/status (server uptime and last reconfiguration time), http://127.0.0.1:8888/json/v1/server (server and resolver statistics), http://127.0.0.1:8888/json/v1/zones (zone statistics), http://127.0.0.1:8888/json/v1/net (network status and socket statistics), http://127.0.0.1:8888/

json/v1/mem (memory manager statistics), http://127.0.0.1:8888/json/v1/tasks (task manager statistics), and http://127.0.0.1:8888/json/v1/traffic (traffic sizes).

4.2.19 `trust-anchors` Statement Grammar

```
trust-anchors { <string> ( static-key |
    initial-key | static-ds | initial-ds )
    <integer> <integer> <integer>
    <quoted_string>; ... };
```

4.2.20 `dnssec-keys` Statement Definition and Usage

The `trust-anchors` statement defines DNSSEC trust anchors. DNSSEC is described in *DNSSEC*.

A trust anchor is defined when the public key or public key digest for a non-authoritative 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 or digest has been configured as a trust anchor, it is treated as if it had been validated and proven secure.

The resolver attempts DNSSEC validation on all DNS data in subdomains of configured trust anchors. Validation below specified names can be temporarily disabled by using `rndc nta`, or permanently disabled with the `validate-except` option.

All keys listed in `trust-anchors`, and their corresponding zones, are deemed to exist regardless of what parent zones say. Only keys configured as trust anchors are used to validate the DNSKEY RRset for the corresponding name. The parent's DS RRset is not used.

`trust-anchors` may be set at the top level of `named.conf` or within a view. If it is set in both places, the configurations are additive: keys defined at the top level are inherited by all views, but keys defined in a view are only used within that view.

The `trust-anchors` statement can contain multiple trust anchor entries, each consisting of a domain name, followed by an "anchor type" keyword indicating the trust anchor's format, followed by the key or digest data.

If the anchor type is `static-key` or `initial-key`, then it is followed with the key's flags, protocol, algorithm, and the Base64 representation of the public key data. This is identical to the text representation of a DNSKEY record. Spaces, tabs, newlines, and carriage returns are ignored in the key data, so the configuration may be split into multiple lines.

If the anchor type is `static-ds` or `initial-ds`, it is followed with the key tag, algorithm, digest type, and the hexadecimal representation of the key digest. This is identical to the text representation of a DS record. Spaces, tabs, newlines, and carriage returns are ignored.

Trust anchors configured with the `static-key` or `static-ds` anchor types are immutable, while keys configured with `initial-key` or `initial-ds` can be kept up-to-date automatically, without intervention from the resolver operator. (`static-key` keys are identical to keys configured using the deprecated `trusted-keys` statement.)

Suppose, for example, that a zone's key-signing key was compromised, and the zone owner had to revoke and replace the key. A resolver which had the original key configured using `static-key` or `static-ds` 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 `trust-anchors` statement with the new key.

If, however, the trust anchor had been configured using `initial-key` or `initial-ds` instead, then the zone owner could add a "stand-by" key to their zone in advance. `named` would store the stand-by key, and when

the original key was revoked, `named` would be able to transition smoothly to the new key. It would also recognize that the old key had been revoked, and cease using that key to validate answers, minimizing the damage that the compromised key could do. This is the process used to keep the ICANN root DNSSEC key up-to-date.

Whereas `static-key` and `static-ds` trust anchors continue to be trusted until they are removed from `named.conf`, an `initial-key` or `initial-ds` is only trusted *once*: for as long as it takes to load the managed key database and start the **RFC 5011** key maintenance process.

It is not possible to mix static with initial trust anchors for the same domain name.

The first time `named` runs with an `initial-key` or `initial-ds` configured in `named.conf`, it fetches the DNSKEY RRset directly from the zone apex, and validates it using the trust anchor specified in `trust-anchors`. If the DNSKEY RRset is validly signed by a key matching the trust anchor, then it is used as the basis for a new managed keys database.

From that point on, whenever `named` runs, it sees the `initial-key` or `initial-ds` listed in `trust-anchors`, checks to make sure **RFC 5011** key maintenance has already been initialized for the specified domain, and if so, simply moves on. The key specified in the `trust-anchors` statement is not used to validate answers; it is superseded by the key or keys stored in the managed keys database.

The next time `named` runs after an `initial-key` or `initial-ds` has been *removed* from the `dnssec-keys` statement (or changed to a `static-key` or `static-ds`), the corresponding zone is removed from the managed keys database, and **RFC 5011** key maintenance is no longer used for that domain.

In the current implementation, the managed keys database is stored as a master-format zone file.

On servers which do not use views, this file is named `managed-keys.bind`. When views are in use, there is a separate managed keys database for each view; the filename is the view name (or, if a view name contains characters which would make it illegal as a filename, 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 are written into a journal file, e.g., `managed-keys.bind.jnl` or `internal.mkeys.jnl`. Changes are committed to the primary file as soon as possible afterward; this usually occurs within 30 seconds. Whenever `named` is using automatic key maintenance, the zone file and journal file can be expected to exist in the working directory. (For this reason, among others, the working directory should be always be writable by `named`.)

If the `dnssec-validation` option is set to `auto`, `named` automatically initializes an `initial-key` for the root zone. The key that is used to initialize the key maintenance process is stored in `bind.keys`; the location of this file can be overridden with the `bindkeys-file` option. As a fallback in the event no `bind.keys` can be found, the initializing key is also compiled directly into `named`.

```
dnssec-policy <string> {
      dnskey-ttl <duration>;
      keys { ( csk | ksk | zsk ) [ ( key-directory ) ] lifetime
          <duration_or_unlimited> algorithm <string> [ <integer> ]; ... };
      max-zone-ttl <duration>;
      parent-ds-ttl <duration>;
      parent-propagation-delay <duration>;
      parent-registration-delay <duration>;
      publish-safety <duration>;
      retire-safety <duration>;
      signatures-refresh <duration>;
      signatures-validity <duration>;
      signatures-validity-dnskey <duration>;
      zone-propagation-delay <duration>;
};
```

The `dnssec-policy` statement defines a key and signing policy (KASP) for zones.

A KASP determines how one or more zones is signed with DNSSEC. For example, it specifies how often keys should roll, which cryptographic algorithms to use, and how often RRSIG records need to be refreshed.

Keys are not shared among zones, which means that one set of keys per zone is generated even if they have the same policy. If multiple views are configured with different versions of the same zone, each separate version uses the same set of signing keys.

Multiple key and signing policies can be configured. To attach a policy to a zone, add a `dnssec-policy` option to the `zone` statement, specifying the name of the policy that should be used.

Key rollover timing is computed for each key according to the key lifetime defined in the KASP. The lifetime may be modified by zone TTLs and propagation delays, to prevent validation failures. When a key reaches the end of its lifetime, `named` generates and publishes a new key automatically, then deactivates the old key and activates the new one, and finally retires the old key according to a computed schedule.

Zone-signing key (ZSK) rollovers require no operator input. Key-signing key (KSK) and combined-signing key (CSK) rollovers require action to be taken to submit a DS record to the parent. Rollover timing for KSKs and CSKs is adjusted to take into account delays in processing and propagating DS updates.

There are two predefined `dnssec-policy` names: `none` and `default`. Setting a zone's policy to `none` is the same as not setting `dnssec-policy` at all; the zone is not signed. Policy `default` causes the zone to be signed with a single combined-signing key (CSK) using algorithm ECDSAP256SHA256; this key has an unlimited lifetime. (A verbose copy of this policy may be found in the source tree, in the file `doc/misc/dnssec-policy.default.conf`.)

Note: The default signing policy may change in future releases. This could require changes to a signing policy when upgrading to a new version of BIND. Check the release notes carefully when upgrading to be informed of such changes. To prevent policy changes on upgrade, use an explicitly defined `dnssec-policy`, rather than `default`.

If a `dnssec-policy` statement is modified and the server restarted or reconfigured, `named` attempts to change the policy smoothly from the old one to the new. For example, if the key algorithm is changed, then a new key is generated with the new algorithm, and the old algorithm is retired when the existing key's lifetime ends.

Note: Rolling to a new policy while another key rollover is already in progress is not yet supported, and may result in unexpected behavior.

The following options can be specified in a `dnssec-policy` statement:

`dnskey-ttl` This indicates the TTL to use when generating DNSKEY resource records. The default is 1 hour (3600 seconds).

`keys` This is a list specifying the algorithms and roles to use when generating keys and signing the zone. Entries in this list do not represent specific DNSSEC keys, which may be changed on a regular basis, but the roles that keys play in the signing policy. For example, configuring a KSK of algorithm RSASHA256 ensures that the DNSKEY RRset always includes a key-signing key for that algorithm.

Here is an example (for illustration purposes only) of some possible entries in a `keys` list:

```
keys {
       ksk key-directory lifetime unlimited algorithm rsasha1 2048;
       zsk lifetime P30D algorithm 8;
       csk lifetime P6MT12H3M15S algorithm ecdsa256;
```

(continues on next page)

(continued from previous page)

```
 };

This example specifies that three keys should be used
in the zone. The first token determines which role the
key plays in signing RRsets.  If set to
``ksk``, then this is
a key-signing key; it has the KSK flag set and
is only used to sign DNSKEY, CDS, and CDNSKEY RRsets.
If set to ``zsk``, this is
a zone-signing key; the KSK flag is unset, and
the key signs all RRsets *except*
DNSKEY, CDS, and CDNSKEY. If set to
``csk``, the key has the KSK
flag set and is used to sign all RRsets.

An optional second token determines where the key is
stored.  Currently, keys can only be stored in the
configured ``key-directory``. This token
may be used in the future to store keys in hardware
service modules or separate directories.

The ``lifetime`` parameter specifies how
long a key may be used before rolling over.  In the
example above, the first key has an unlimited
lifetime, the second key may be used for 30 days, and the
third key has a rather peculiar lifetime of 6 months,
12 hours, 3 minutes, and 15 seconds.  A lifetime of 0
seconds is the same as ``unlimited``.

Note that the lifetime of a key may be extended if
retiring it too soon would cause validation failures.
For example, if the key were configured to roll more
frequently than its own TTL, its lifetime would
automatically be extended to account for this.

The ``algorithm`` parameter specifies
the key's algorithm, expressed either as a string
("rsasha256", "ecdsa384", etc.) or as a decimal number.
An optional second parameter specifies the key's size
in bits. If it is omitted, as shown in the
example for the second and third keys, an appropriate
default size for the algorithm is used.
```

publish-safety

This is a margin that is added to the pre-publication interval in rollover timing calculations, to give some extra time to cover unforeseen events. This increases the time between when keys are published and they become active. The default is `PT1H` (1 hour).

retire-safety This is a margin that is added to the post-publication interval in rollover timing calculations, to give some extra time to cover unforeseen events. This increases the time a key remains published after it is no longer active. The default is `PT1H` (1 hour).

signatures-refresh This determines how frequently an RRSIG record needs to be refreshed. The signature is renewed when the time until the expiration time is closer than the specified interval. The default is `P5D` (5 days), meaning signatures that expire in 5 days or sooner are refreshed.

signatures-validity This indicates the validity period of an RRSIG record (subject to inception offset and jitter). The default is `P2W` (2 weeks).

signatures-validity-dnskey This is similar to **signatures-validity**, but for DNSKEY records. The default is `P2W` (2 weeks).

max-zone-ttl Like the `max-zone-ttl` zone option, this specifies the maximum permissible TTL value in seconds for the zone. When loading a zone file using a *masterfile-format* of `text` or `raw`, any record encountered with a TTL higher than *max-zone-ttl* is capped at the maximum permissible TTL value.

This is needed in DNSSEC-maintained zones because when rolling to a new DNSKEY, the old key needs to remain available until RRSIG records have expired from caches. The *max-zone-ttl* option guarantees that the largest TTL in the zone is no higher than the set value.

Note: Because `map`-format files load directly into memory, this option cannot be used with them.

The default value is `PT24H` (24 hours). A *max-zone-ttl* of zero is treated as if the default value were in use.

zone-propagation-delay This is the expected propagation delay from the time when a zone is first updated to the time when the new version of the zone is served by all secondary servers. The default is `PT5M` (5 minutes).

parent-ds-ttl This is the TTL of the DS RRset that the parent zone uses. The default is `P1D` (1 day).

parent-propagation-delay This is the expected propagation delay from the time when the parent zone is updated to the time when the new version is served by all of the parent zone's name servers. The default is `PT1H` (1 hour).

parent-registration-delay This is the expected registration delay from the time when a DS RRset change is requested to the time when the DS RRset is updated in the parent zone. The default is `P1D` (1 day).

4.2.21 `managed-keys` Statement Grammar

```
managed-keys { <string> ( static-key
    | initial-key | static-ds |
    initial-ds ) <integer> <integer>
    <integer> <quoted_string>; ... };, deprecated
```

4.2.22 `managed-keys` Statement Definition and Usage

The `managed-keys` statement has been deprecated in favor of *trust-anchors Statement Grammar* with the `initial-key` keyword.

4.2.23 `trusted-keys` Statement Grammar

```
trusted-keys { <string> <integer>
    <integer> <integer>
    <quoted_string>; ... };, deprecated
```

4.2.24 `trusted-keys` Statement Definition and Usage

The `trusted-keys` statement has been deprecated in favor of *trust-anchors Statement Grammar* with the `static-key` keyword.

4.2.25 `view` Statement Grammar

```
view view_name [ class ] {
    match-clients { address_match_list } ;
    match-destinations { address_match_list } ;
    match-recursive-only yes_or_no ;
  [ view_option ; ... ]
  [ zone_statement ; ... ]
} ;
```

4.2.26 `view` Statement Definition and Usage

The `view` statement is a powerful feature of BIND 9 that lets a name server 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 is seen by a subset of clients. A client matches a view if its source IP address matches the `address_match_list` of the view's `match-clients` clause, and its destination IP address matches the `address_match_list` of the view's `match-destinations` clause. If not specified, both `match-clients` and `match-destinations` default to matching all addresses. In addition to checking IP addresses, `match-clients` and `match-destinations` can also take `keys` which provide an mechanism for the client to select the view. A view can also be specified as `match-recursive-only`, which means that only recursive requests from matching clients will match that view. The order of the `view` statements is significant — a client request is resolved in the context of the first `view` that it matches.

Zones defined within a `view` statement are only accessible to clients that match the `view`. By defining a zone of the same name in multiple views, different zone data can be given 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 then apply only when resolving 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; these view-specific defaults take precedence over those in the `options` statement.

Views are class-specific. If no class is given, class IN is assumed. Note that all non-IN views must contain a hint zone, since only the IN class has compiled-in default hints.

If there are no `view` statements in the config file, a default view that matches any client is automatically created in class IN. Any `zone` statements specified on the top level of the configuration file are considered to be part of this default view, and the `options` statement applies to the default view. If any explicit `view` statements are present, all `zone` statements must occur inside `view` statements.

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";
      };
};

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";
      };
};
```

4.2.27 `zone` Statement Grammar

```
zone <string> [ <class> ] {
      type ( master | primary );
      allow-query { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      allow-transfer { <address_match_element>; ... };
      allow-update { <address_match_element>; ... };
      also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ 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 );
```

(continues on next page)

(continued from previous page)

```
      check-mx-cname ( fail | warn | ignore );
      check-names ( fail | warn | ignore );
      check-sibling <boolean>;
      check-spf ( warn | ignore );
      check-srv-cname ( fail | warn | ignore );
      check-wildcard <boolean>;
      database <string>;
      dialup ( notify | notify-passive | passive | refresh | <boolean> );
      dlz <string>;
      dnskey-sig-validity <integer>;
      dnssec-dnskey-kskonly <boolean>;
      dnssec-loadkeys-interval <integer>;
      dnssec-policy <string>;
      dnssec-secure-to-insecure <boolean>;
      dnssec-update-mode ( maintain | no-resign );
      file <quoted_string>;
      forward ( first | only );
      forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ] [ dscp <integer> ]; ... };
      inline-signing <boolean>;
      ixfr-from-differences <boolean>;
      journal <quoted_string>;
      key-directory <quoted_string>;
      masterfile-format ( map | raw | text );
      masterfile-style ( full | relative );
      max-journal-size ( default | unlimited | <sizeval> );
      max-records <integer>;
      max-transfer-idle-out <integer>;
      max-transfer-time-out <integer>;
      max-zone-ttl ( unlimited | <duration> );
      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>;
      serial-update-method ( date | increment | unixtime );
      sig-signing-nodes <integer>;
      sig-signing-signatures <integer>;
      sig-signing-type <integer>;
      sig-validity-interval <integer> [ <integer> ];
      update-check-ksk <boolean>;
      update-policy ( local | { ( deny | grant ) <string> ( 6to4-self | external | krb5-
↪self | krb5-selfsub | krb5-subdomain | ms-self | ms-selfsub | ms-subdomain | name |␣
↪self | selfsub | selfwild | subdomain | tcp-self | wildcard | zonesub ) [ <string> ]
↪<rrtypelist>; ... };
      zero-no-soa-ttl <boolean>;
      zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
```

(continues on next page)

(continued from previous page)

```
     type ( slave | secondary );
     allow-notify { <address_match_element>; ... };
     allow-query { <address_match_element>; ... };
     allow-query-on { <address_match_element>; ... };
     allow-transfer { <address_match_element>; ... };
     allow-update-forwarding { <address_match_element>; ... };
     also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ 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-names ( fail | warn | ignore );
     database <string>;
     dialup ( notify | notify-passive | passive | refresh | <boolean> );
     dlz <string>;
     dnskey-sig-validity <integer>;
     dnssec-dnskey-kskonly <boolean>;
     dnssec-loadkeys-interval <integer>;
     dnssec-policy <string>;
     dnssec-update-mode ( maintain | no-resign );
     file <quoted_string>;
     forward ( first | only );
     forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ] [ dscp <integer> ]; ... };
     inline-signing <boolean>;
     ixfr-from-differences <boolean>;
     journal <quoted_string>;
     key-directory <quoted_string>;
     masterfile-format ( map | raw | text );
     masterfile-style ( full | relative );
     masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
     max-journal-size ( default | unlimited | <sizeval> );
     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>;
     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>;
     request-expire <boolean>;
```

(continues on next page)

(continued from previous page)

```
      request-ixfr <boolean>;
      sig-signing-nodes <integer>;
      sig-signing-signatures <integer>;
      sig-signing-type <integer>;
      sig-validity-interval <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> );
};
```

```
zone <string> [ <class> ] {
      type mirror;
      allow-notify { <address_match_element>; ... };
      allow-query { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      allow-transfer { <address_match_element>; ... };
      allow-update-forwarding { <address_match_element>; ... };
      also-notify [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
      alt-transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp
↪<integer> ];
      alt-transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp
↪<integer> ];
      check-names ( fail | warn | ignore );
      database <string>;
      file <quoted_string>;
      ixfr-from-differences <boolean>;
      journal <quoted_string>;
      masterfile-format ( map | raw | text );
      masterfile-style ( full | relative );
      masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
      max-journal-size ( default | unlimited | <sizeval> );
      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>;
      min-refresh-time <integer>;
      min-retry-time <integer>;
      multi-master <boolean>;
      notify ( explicit | master-only | <boolean> );
      notify-delay <integer>;
```

(continues on next page)

(continued from previous page)

```
      notify-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer> ];
      notify-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer>
↪ ];
      request-expire <boolean>;
      request-ixfr <boolean>;
      transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer>␣
↪];
      transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp
↪<integer> ];
      try-tcp-refresh <boolean>;
      use-alt-transfer-source <boolean>;
      zero-no-soa-ttl <boolean>;
      zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
      type hint;
      check-names ( fail | warn | ignore );
      delegation-only <boolean>;
      file <quoted_string>;
};
```

```
zone <string> [ <class> ] {
      type stub;
      allow-query { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      check-names ( fail | warn | ignore );
      database <string>;
      delegation-only <boolean>;
      dialup ( notify | notify-passive | passive | refresh | <boolean> );
      file <quoted_string>;
      forward ( first | only );
      forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ] [ dscp <integer> ]; ... };
      masterfile-format ( map | raw | text );
      masterfile-style ( full | relative );
      masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
      max-records <integer>;
      max-refresh-time <integer>;
      max-retry-time <integer>;
      max-transfer-idle-in <integer>;
      max-transfer-time-in <integer>;
      min-refresh-time <integer>;
      min-retry-time <integer>;
      multi-master <boolean>;
      transfer-source ( <ipv4_address> | * ) [ port ( <integer> | * ) ] [ dscp <integer>␣
↪];
      transfer-source-v6 ( <ipv6_address> | * ) [ port ( <integer> | * ) ] [ dscp
↪<integer> ];
      use-alt-transfer-source <boolean>;
```

(continues on next page)

(continued from previous page)

```
      zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
      type static-stub;
      allow-query { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      forward ( first | only );
      forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ] [ dscp <integer> ]; ... };
      max-records <integer>;
      server-addresses { ( <ipv4_address> | <ipv6_address> ); ... };
      server-names { <string>; ... };
      zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
      type forward;
      delegation-only <boolean>;
      forward ( first | only );
      forwarders [ port <integer> ] [ dscp <integer> ] { ( <ipv4_address> | <ipv6_
↪address> ) [ port <integer> ] [ dscp <integer> ]; ... };
};
```

```
zone <string> [ <class> ] {
      type redirect;
      allow-query { <address_match_element>; ... };
      allow-query-on { <address_match_element>; ... };
      dlz <string>;
      file <quoted_string>;
      masterfile-format ( map | raw | text );
      masterfile-style ( full | relative );
      masters [ port <integer> ] [ dscp <integer> ] { ( <masters> | <ipv4_address> [␣
↪port <integer> ] | <ipv6_address> [ port <integer> ] ) [ key <string> ]; ... };
      max-records <integer>;
      max-zone-ttl ( unlimited | <duration> );
      zone-statistics ( full | terse | none | <boolean> );
};
```

```
zone <string> [ <class> ] {
      type delegation-only;
};
```

```
zone <string> [ <class> ] {
      in-view <string>;
};
```

4.2.28 `zone` Statement Definition and Usage

Zone Types

The `type` keyword is required for the `zone` configuration unless it is an `in-view` configuration. Its acceptable values include: `primary` (or `master`), `secondary` (or `slave`), `mirror`, `delegation-only`, `forward`, `hint`, `redirect`, `static-stub`, and `stub`.

`primary` A primary zone has a master copy of the data for the zone and is able to provide authoritative answers for it. Type `master` is a synonym for `primary`.

`secondary` A secondary zone is a replica of a primary zone. Type `slave` is a synonym for `secondary`. The `masters` list specifies one or more IP addresses of primary servers that the secondary 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 primary can also be done with per-server TSIG keys. If a file is specified, then the replica is 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 secondary 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 there are 100000 files in a single directory.)

`stub` A stub zone is similar to a secondary zone, except that it replicates only the NS records of a primary zone instead of the entire zone. Stub zones are not a standard part of the DNS; they are a feature specific to the BIND implementation.

Stub zones can be used to eliminate the need for a 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 BIND 9 supports it only in a limited way. If a BIND 9 primary, serving a parent zone, has child stub zones configured, all the secondary 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 **RFC 1918** 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.

`mirror`

A mirror zone is similar to a zone of type `secondary`, except its data is subject to DNSSEC validation before being used in answers. Validation is applied to the entire zone during the zone transfer process, and again when the zone file is loaded from disk upon restarting `named`. If validation of a new version of a mirror zone fails, a retransfer is scheduled and the most recent correctly validated version of that zone is used, until it either expires or a newer version validates correctly. If no usable zone data is available for a mirror zone, either due to transfer failure or expiration, traditional DNS recursion is used to look up the answers instead. Mirror zones cannot be used in a view that does not have recursion enabled.

Answers coming from a mirror zone look almost exactly like answers from a zone of type `secondary`, with the notable exceptions that the AA bit ("authoritative answer") is not set, and the AD bit ("authenticated data") is.

Mirror zones are intended to be used to set up a fast local copy of the root zone, similar to the one described in **RFC 7706**. A default list of primary servers for the IANA root zone is built into `named` and thus its mirroring can be enabled using the following configuration:

```
zone "." {
     type mirror;
};
```

Mirroring a zone other than root requires an explicit list of primary servers to be provided using the `masters` option (see *masters Statement Grammar* for details), and a key-signing key (KSK) for the specified zone to be explicitly configured as a trust anchor.

To make mirror zone contents persist between `named` restarts, use the *file* option.

When configuring NOTIFY for a mirror zone, only `notify no;` and `notify explicit;` can be used at the zone level. Using any other `notify` setting at the zone level is a configuration error. Using any other `notify` setting at the `options` or `view` level causes that setting to be overridden with `notify explicit;` for the mirror zone. The global default for the `notify` option is `yes`, so mirror zones are by default configured with `notify explicit;`.

Outgoing transfers of mirror zones are disabled by default but may be enabled using *allow-transfer*.

Note: Using this zone type with any zone other than the root zone should be considered *experimental* and may cause performance issues, especially for zones which are large and/or frequently updated.

`static-stub` 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 primary server; and when recursion is necessary for a query that matches a static-stub zone, the locally configured data (name server names and glue addresses) are 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 with `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 are responded to 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 is rejected with an rcode of NOTAUTH.

Each static-stub zone is configured with internally generated NS and (if necessary) glue A or AAAA RRs.

`forward` 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 `forwarders` statement, which applies to queries within the domain given by the zone name. If no `forwarders` statement is present, or an empty list for `forwarders` is given, then no forwarding is done for the domain, canceling the effects of any forwarders in the `options` statement. Thus, to use this type of zone to change the behavior of the global `forward` option (that is, "forward first" to, then "forward only", or vice versa), but using the same servers as set globally, re-specify the global forwarders.

`hint` The initial set of root name servers is specified using a hint zone. When the server starts, it uses the root hints to find a root name server and get the most recent list of root name servers. If no hint zone is specified for class IN, the server uses a compiled-in default set of root servers hints. Classes other than IN have no built-in default hints.

`redirect` Redirect zones are used to provide answers to queries when normal resolution would result in NXDOMAIN being returned. Only one redirect zone is supported per view. `allow-query` can be used

to restrict which clients see these answers.

If the client has requested DNSSEC records (DO=1) and the NXDOMAIN response is signed, no substitution occurs.

To redirect all NXDOMAIN responses to 100.100.100.2 and 2001:ffff:ffff::100.100.100.2, configure a type `redirect` zone named ".", with the zone file containing wildcard records that point to the desired addresses: `"*. IN A 100.100.100.2"` and `"*. IN AAAA 2001:ffff:ffff::100.100.100.2"`.

As another example, to redirect all Spanish names (under .ES), use similar entries but with the names `*.ES.` instead of `*.`. To redirect all commercial Spanish names (under COM.ES), use wildcard entries called `*.COM.ES.`.

Note that the redirect zone supports all possible types; it is not limited to A and AAAA records.

If a redirect zone is configured with a `masters` option, then it is transferred in as if it were a secondary zone. Otherwise, it is loaded from a file as if it were a primary zone.

Because redirect zones are not referenced directly by name, they are not kept in the zone lookup table with normal primary and secondary zones. To reload a redirect zone, use `rndc reload -redirect`; to retransfer a redirect zone configured as a secondary, use `rndc retransfer -redirect`. When using `rndc reload` without specifying a zone name, redirect zones are reloaded along with other zones.

`delegation-only` This zone type is used to enforce the delegation-only status of infrastructure zones (e.g., COM, NET, ORG). Any answer that is received without an explicit or implicit delegation in the authority section is treated as NXDOMAIN. This does not apply to the zone apex, and should not be applied to leaf zones.

`delegation-only` has no effect on answers received from forwarders.

See caveats in *root-delegation-only*.

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 was 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.

Zone Options

`allow-notify` See the description of `allow-notify` in *Access Control*.

`allow-query` See the description of `allow-query` in *Access Control*.

`allow-query-on` See the description of `allow-query-on` in *Access Control*.

`allow-transfer` See the description of `allow-transfer` in *Access Control*.

`allow-update` See the description of `allow-update` in *Access Control*.

`update-policy` This specifies a "Simple Secure Update" policy. See *Dynamic Update Policies*.

`allow-update-forwarding` See the description of `allow-update-forwarding` in *Access Control*.

also-notify This option is only meaningful if `notify` is active for this zone. The set of machines that receive a `DNS NOTIFY` message for this zone is made up of all the listed name servers (other than the primary) 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 primary files and/or DNS responses received from the network. The default varies according to zone type. For `primary` zones the default is `fail`. For `secondary` zones the default is `warn`. It is not implemented for `hint` zones.

check-mx See the description of `check-mx` in *Boolean Options*.

check-spf See the description of `check-spf` in *Boolean Options*.

check-wildcard See the description of `check-wildcard` in *Boolean Options*.

check-integrity See the description of `check-integrity` in *Boolean Options*.

check-sibling See the description of `check-sibling` in *Boolean Options*.

zero-no-soa-ttl See the description of `zero-no-soa-ttl` in *Boolean Options*.

update-check-ksk See the description of `update-check-ksk` in *Boolean Options*.

dnssec-loadkeys-interval See the description of `dnssec-loadkeys-interval` in *options Statement Definition and Usage*.

dnssec-update-mode See the description of `dnssec-update-mode` in *options Statement Definition and Usage*.

dnssec-dnskey-kskonly See the description of `dnssec-dnskey-kskonly` in *Boolean Options*.

try-tcp-refresh See the description of `try-tcp-refresh` in *Boolean Options*.

database This specifies 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"`, BIND 9'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 *Boolean Options*.

delegation-only This flag only applies to forward, hint, and stub zones. If set to `yes`, then the zone is treated as if it is also a delegation-only type zone.

See caveats in *root-delegation-only*.

file This sets the zone's filename. In `primary`, `hint`, and `redirect` zones which do not have `masters` defined, zone data is loaded from this file. In `secondary`, `mirror`, `stub`, and `redirect` 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 This option is 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` allows a normal lookup to be tried.

forwarders This is 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 This allows the default journal's filename to be overridden. The default is the zone's filename with ".jnl" appended. This is applicable to **primary** and **secondary** zones.

max-ixfr-ratio See the description of **max-ixfr-ratio** in *options Statement Definition and Usage*.

max-journal-size See the description of **max-journal-size** in *Server Resource Limits*.

max-records See the description of **max-records** in *Server Resource Limits*.

max-transfer-time-in See the description of **max-transfer-time-in** in *Zone Transfers*.

max-transfer-idle-in See the description of **max-transfer-idle-in** in *Zone Transfers*.

max-transfer-time-out See the description of **max-transfer-time-out** in *Zone Transfers*.

max-transfer-idle-out See the description of **max-transfer-idle-out** in *Zone Transfers*.

notify See the description of **notify** in *Boolean Options*.

notify-delay See the description of **notify-delay** in *Tuning*.

notify-to-soa See the description of **notify-to-soa** in *Boolean Options*.

zone-statistics See the description of **zone-statistics** in *options Statement Definition and Usage*.

server-addresses This option is 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 internally configures 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 are internally configured:

```
example.com. NS example.com.
example.com. A 192.0.2.1
example.com. AAAA 2001:db8::1234
```

These records are used internally 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 initiates recursive resolution and sends queries to 192.0.2.1 and/or 2001:db8::1234.

server-names This is 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 are 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 subdomain of the origin name of the 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 is rejected by the configuration parser.

A non-empty list for this option internally configures 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 are internally configured:

```
example.com. NS ns1.example.net.
example.com. NS ns2.example.net.
```

These records are used internally 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 initiates recursive resolution, resolves "ns1.example.net" and/or "ns2.example.net" to IP addresses, and then sends queries to one or more of these addresses.

sig-validity-interval See the description of **sig-validity-interval** in *Tuning*.

sig-signing-nodes See the description of **sig-signing-nodes** in *Tuning*.

sig-signing-signatures See the description of **sig-signing-signatures** in *Tuning*.

sig-signing-type See the description of **sig-signing-type** in *Tuning*.

transfer-source See the description of **transfer-source** in *Zone Transfers*.

transfer-source-v6 See the description of **transfer-source-v6** in *Zone Transfers*.

alt-transfer-source See the description of **alt-transfer-source** in *Zone Transfers*.

alt-transfer-source-v6 See the description of **alt-transfer-source-v6** in *Zone Transfers*.

use-alt-transfer-source See the description of **use-alt-transfer-source** in *Zone Transfers*.

notify-source See the description of **notify-source** in *Zone Transfers*.

notify-source-v6 See the description of **notify-source-v6** in *Zone Transfers*.

min-refresh-time; max-refresh-time; min-retry-time; max-retry-time See the descriptions in *Tuning*.

ixfr-from-differences See the description of **ixfr-from-differences** in *Boolean Options*. (Note that the **ixfr-from-differences** choices of **primary** and **secondary** are not available at the zone level.)

key-directory See the description of **key-directory** in *options Statement Definition and Usage*.

auto-dnssec See the description of **auto-dnssec** in *options Statement Definition and Usage*.

serial-update-method See the description of **serial-update-method** in *options Statement Definition and Usage*.

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 behavior is disabled by default.

multi-master See the description of **multi-master** in *Boolean Options*.

masterfile-format See the description of **masterfile-format** in *Tuning*.

max-zone-ttl See the description of **max-zone-ttl** in *options Statement Definition and Usage*.

dnssec-secure-to-insecure See the description of **dnssec-secure-to-insecure** in *Boolean Options*.

Dynamic Update Policies

BIND 9 supports two methods of granting clients the right to perform dynamic updates to a zone, configured by the **allow-update** or **update-policy** options.

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 which 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 the client source address.

update-policy rules are only meaningful for zones of type **primary**, 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;**. **named** automatically generates a TSIG session key when starting and stores 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/named/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 is 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 are permitted to send updates.

Note that only one session key is generated by named; all zones configured to use update-policy local 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 13 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 represensts 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 is 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 16 values: name, subdomain, wildcard, self, selfsub, selfwild, krb5-self, ms-self, krb5-selfsub, ms-selfsub, krb5-subdomain, ms-subdomain, tcp-self, 6to4-self, zonesub, and external.

name With 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 `*` (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.

`ms-self` When a client sends an UPDATE using a Windows machine principal (for example, `machine$@REALM`), this rule allows records with the absolute name of `machine.REALM` to be updated.

The realm to be matched is specified in the identity field.

The name field has no effect on this rule; it should be set to "." as a placeholder.

For example, `grant EXAMPLE.COM ms-self . A AAAA` allows any machine with a valid principal in the realm `EXAMPLE.COM` to update its own address records.

`ms-selfsub` This is similar to `ms-self`, except it also allows updates to any subdomain of the name specified in the Windows machine principal, not just to the name itself.

`ms-subdomain` When a client sends an UPDATE using a Windows machine principal (for example, `machine$@REALM`), this rule allows any machine in the specified realm to update any record in the zone or in a specified subdomain of the zone.

The realm to be matched is specified in the identity field.

The name field specifies the subdomain that may be updated. If set to "." or any other name at or above the zone apex, any name in the zone can be updated.

For example, if `update-policy` for the zone "example.com" includes `grant EXAMPLE.COM ms-subdomain hosts.example.com. AA AAAA`, any machine with a valid principal in the realm `EXAMPLE.COM` is able to update address records at or below `hosts.example.com`.

`krb5-self` When a client sends an UPDATE using a Kerberos machine principal (for example, `host/machine@REALM`), this rule allows records with the absolute name of `machine` to be updated, provided it has been authenticated by REALM. This is similar but not identical to `ms-self`, due to the `machine` part of the Kerberos principal being an absolute name instead of a unqualified name.

The realm to be matched is specified in the identity field.

The name field has no effect on this rule; it should be set to "." as a placeholder.

For example, `grant EXAMPLE.COM krb5-self . A AAAA` allows any machine with a valid principal in the realm `EXAMPLE.COM` to update its own address records.

`krb5-selfsub` This is similar to `krb5-self`, except it also allows updates to any subdomain of the name specified in the `machine` part of the Kerberos principal, not just to the name itself.

`krb5-subdomain` This rule is identical to `ms-subdomain`, except that it works with Kerberos machine principals (i.e., `host/machine@REALM`) rather than Windows machine principals.

`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 to be signed.

Note: It is theoretically possible to spoof these TCP sessions.

`6to4-self` This allows the name matching a 6to4 IPv6 prefix, as specified in **RFC 3056**, to be updated by any TCP connection from either the 6to4 network or from the corresponding IPv4 address. This is intended to allow NS or DNAME RRsets to be added to the `ip6.arpa` reverse tree.

The `identity` field must match the 6to4 prefix in `ip6.arpa`. 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 to be signed.

In addition, if specified for an `ip6.arpa` name outside of the `2.0.0.2.ip6.arpa` namespace, the corresponding /48 reverse name can be updated. For example, TCP/IPv6 connections from 2001:DB8:ED0C::/48 can update records at `C.0.D.E.8.B.D.0.1.0.0.2.ip6.arpa`.

Note: It is theoretically possible to spoof these TCP sessions.

`external` This rule allows `named` to defer the decision of whether to allow a given update to an external daemon.

The method of communicating with the daemon is specified in the identity field, the format of which is "`local:`path", where "path" is the location of a Unix-domain socket. (Currently, "local" is the only supported mechanism.)

Requests to the external daemon are sent over the Unix-domain socket as datagrams with the following format:

```
Protocol version number (4 bytes, network byte order, currently 1)
Request length (4 bytes, network byte order)
Signer (null-terminated string)
Name (null-terminated string)
TCP source address (null-terminated string)
Rdata type (null-terminated string)
Key (null-terminated string)
TKEY token length (4 bytes, network byte order)
TKEY token (remainder of packet)
```

The daemon replies with a four-byte value in network byte order, containing either 0 or 1; 0 indicates that the specified update is not permitted, and 1 indicates that it is.

Multiple views

When multiple views are in use, a zone may be referenced by more than one of them. Often, the views 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. For example:

```
view internal {
    match-clients { 10/8; };

    zone example.com {
    type master;
```

(continues on next page)

(continued from previous page)

```
    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 change 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 needs 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.

4.3 Zone File

4.3.1 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 type is used. Since the publication of **RFC 1034**, several new RRs have been identified and implemented in the DNS. These are also included.

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 *The sortlist Statement* and *RRset Ordering*.

The components of a Resource Record are:

owner 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.

RDATA The resource data. The format of the data is type- and sometimes class-specific.

For a complete list of *types* of valid RRs, including those that have been obsoleted, please refer to https://en.wikipedia.org/wiki/List_of_DNS_record_types.

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. It was rarely used for its historical purpose, but was reused for BIND's built-in server information zones, e.g., `version.bind`.

HS Hesiod, an information service developed by MIT's Project Athena. It was used to share information about various systems databases, such as users, groups, printers, etc.

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 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 is anticipated, the TTL can be reduced prior to the change to minimize inconsistency, 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.

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 primary 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 are listed 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. 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 interest of clarity.

The resource data or RDATA section of the RR is given using knowledge of the typical representation for the data.

For example, the RRs carried in a message might be shown 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.

Here's another possible example:

XX.LCS.MIT.EDU.	IN A	10.0.0.44
	CH A	MIT.EDU. 2420

This shows two addresses for `XX.LCS.MIT.EDU`, each of a different class.

4.3.2 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 an 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 falls back to the next largest priority. Priority numbers do not have any absolute meaning; they are relevant only respective to other MX records for that domain name. The domain name given is the machine to which the mail is 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 is ignored. Instead, the mail is 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 is attempted to `mail.example.com` and `mail2.example.com` (in any order); if neither of those succeed, delivery to `mail.backup.org` is attempted.

4.3.3 Setting TTLs

The time-to-live of the RR field 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 is the negative caching TTL. This controls how long other servers cache no-such-domain (NXDOMAIN) responses from this server.

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 controls 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`.

4.3.4 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` line in this example is only to provide context; it does not necessarily appear in the actual usage. It is only used here to indicate that the example is relative to the listed origin.

4.3.5 Other Zone File Directives

The 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 class.

Master File Directives include `$ORIGIN`, `$INCLUDE`, and `$TTL`.

The `@` (at-sign)

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 a trailing dot (.).

The `$ORIGIN` Directive

Syntax: `$ORIGIN` domain-name [comment]

`$ORIGIN` sets the domain name that is appended to any unqualified records. When a zone is first read, there is an implicit `$ORIGIN` <`zone_name`>., followed by a 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.
```

The `$INCLUDE` Directive

Syntax: `$INCLUDE` filename [origin] [comment]

This reads and processes the file `filename` as if it were included in 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. BIND 9 restores both of them. This could be construed as a deviation from **RFC 1035**, a feature, or both.

The `$TTL` Directive

Syntax: `$TTL` default-ttl [comment]

This sets 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**.

4.3.6 BIND Master File Extension: the `$GENERATE` Directive

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**.

```
$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.
```

Both generate a set of A and MX records. Note the MX's right-hand side is a quoted string. The quotes are 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 .
```

(continues on next page)

(continued from previous page)

```
...
HOST-127.EXAMPLE. A  1.2.3.127
HOST-127.EXAMPLE. MX 0 .
```

range 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".

owner This describes the owner name of the resource records to be created. Any single `$` (dollar sign) symbols within the `owner` string are replaced by the iterator value. To get a `$` in the output, 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 `$` as in `${offset[,width[,base]]}`. For example, `${-20,3,d}` subtracts 20 from the current value and 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 `${0,0,d}`. If the `owner` is not absolute, the current `$ORIGIN` is appended to the name.

In nibble mode, the value is treated as if it were a reversed hexadecimal string, with each hexadecimal digit as a separate label. The width field includes the label separator.

For compatibility with earlier versions, `$$` is still recognized as indicating a literal $ in the output.

ttl This specifies the time-to-live of the generated records. If not specified, this is inherited using the normal TTL inheritance rules.

`class` and `ttl` can be entered in either order.

class This specifies the class of the generated records. This must match the zone class if it is specified.

`class` and `ttl` can be entered in either order.

type This can be any valid type.

rdata This is a string containing the RDATA of the resource record to be created. It may be quoted if there are spaces in the string; the quotation marks do not appear in the generated record.

The `$GENERATE` directive is a BIND extension and not part of the standard zone file format.

4.3.7 Additional File Formats

In addition to the standard text format, BIND 9 supports the ability to read or dump to zone files in other formats.

The `raw` format is a binary representation of zone data in a manner similar to that used in zone transfers. Since it does not require parsing text, load time is significantly reduced.

An even faster alternative is the `map` format, which is an image of a BIND 9 in-memory zone database; it can be loaded directly into memory via the `mmap()` function and the zone can begin serving queries almost immediately.

For a primary server, a zone file in `raw` or `map` format is expected to be generated from a textual zone file by the `named-compilezone` command. For a secondary server or a dynamic zone, the zone file is automatically generated when `named` dumps the zone contents after zone transfer or when applying prior updates, if one of these formats is specified by the `masterfile-format` option.

If a zone file in a binary format needs manual modification, it first must be converted to a textual form by the `named-compilezone` command. Make any necessary modifications to the text file, and then convert it to the binary form via the `named-compilezone` command again.

Note that `map` format is extremely architecture-specific. A `map` file *cannot* be used on a system with different pointer size, endianness, or data alignment than the system on which it was generated, and should in general be used only inside a single system. While `raw` format uses network byte order and avoids architecture-dependent data alignment so that it is as portable as possible, it is also primarily expected to be used inside the same single system. To export a zone file in either `raw` or `map` format, or make a portable backup of such a file, conversion to `text` format is recommended.

4.4 BIND 9 Statistics

BIND 9 maintains lots of statistics information and provides several interfaces for users to access those statistics. The available statistics include all statistics counters that are meaningful in BIND 9, and other information that is considered useful.

The statistics information is categorized into the following sections:

Incoming Requests The number of incoming DNS requests for each OPCODE.

Incoming Queries The number of incoming queries for each RR type.

Outgoing Queries The number of outgoing queries for each RR type sent from the internal resolver, maintained per view.

Name Server Statistics Statistics counters for incoming request processing.

Zone Maintenance Statistics Statistics counters regarding zone maintenance operations, such as zone transfers.

Resolver Statistics Statistics counters for name resolutions performed in the internal resolver, maintained per view.

Cache DB RRsets Statistics counters related to cache contents, maintained per view.

The "NXDOMAIN" counter is the number of names that have been cached as nonexistent. Counters named for RR types indicate the number of active RRsets for each type in the cache database.

If an RR type name is preceded by an exclamation mark (!), it represents the number of records in the cache which indicate that the type does not exist for a particular name (this is also known as "NXRRSET"). If an RR type name is preceded by a hash mark (#), it represents the number of RRsets for this type that are present in the cache but whose TTLs have expired; these RRsets may only be used if stale answers are enabled. If an RR type name is preceded by a tilde (~), it represents the number of RRsets for this type that are present in the cache database but are marked for garbage collection; these RRsets cannot be used.

Socket I/O Statistics Statistics counters for network-related events.

A subset of Name Server 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 *options Statement Definition and Usage* 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.

There are currently two user interfaces to get access to the statistics. One is in the plain text format dumped to the file specified by the `statistics-file` configuration option; the other is remotely accessible via a statistics channel when the `statistics-channels` statement is specified in the configuration file (see *statistics-channels Statement Grammar*.)

4.4.1 The Statistics File

The text format statistics dump begins with a line, like:

```
+++ Statistics Dump +++ (973798949)
```

The number in parentheses is a standard Unix-style timestamp, measured in seconds since January 1, 1970. Following that line is a set of statistics information, which is categorized as described above. Each section begins with a line, like:

```
++ Name Server Statistics ++
```

Each section consists of lines, each containing the statistics counter value followed by its textual description; see below for available counters. For brevity, counters that have a value of 0 are not shown in the statistics file.

The statistics dump ends with the line where the number is identical to the number in the beginning line; for example:

```
--- Statistics Dump --- (973798949)
```

4.4.2 Statistics Counters

The following lists summarize the statistics counters that BIND 9 provides. For each counter, the abbreviated symbol name is given; these symbols are shown in the statistics information accessed via an HTTP statistics channel. The description of the counter is also shown in the statistics file but, in this document, may be slightly modified for better readability.

Name Server Statistics Counters

Requestv4 This indicates the number of IPv4 requests received. Note: this also counts non-query requests.

Requestv6 This indicates the number of IPv6 requests received. Note: this also counts non-query requests.

ReqEdns0 This indicates the number of requests received with EDNS(0).

ReqBadEDN SVer This indicates the number of requests received with an unsupported EDNS version.

ReqTSIG This indicates the number of requests received with TSIG.

ReqSIG0 This indicates the number of requests received with SIG(0).

ReqBadSIG This indicates the number of requests received with an invalid (TSIG or SIG(0)) signature.

ReqTCP This indicates the number of TCP requests received.

AuthQryRej This indicates the number of rejected authoritative (non-recursive) queries.

RecQryRej This indicates the number of rejected recursive queries.

XfrRej This indicates the number of rejected zone transfer requests.

UpdateRej This indicates the number of rejected dynamic update requests.

Response This indicates the number of responses sent.

RespTruncated This indicates the number of truncated responses sent.

RespEDNS0 This indicates the number of responses sent with EDNS(0).

RespTSIG This indicates the number of responses sent with TSIG.

RespSIG0 This indicates the number of responses sent with SIG(0).

QrySuccess This indicates the number of queries that resulted in a successful answer, meaning queries which return a NOERROR response with at least one answer RR. This corresponds to the `success` counter of previous versions of BIND 9.

QryAuthAns This indicates the number of queries that resulted in an authoritative answer.

QryNoauthAns This indicates the number of queries that resulted in a non-authoritative answer.

QryReferral This indicates the number of queries that resulted in a referral answer. This corresponds to the `referral` counter of previous versions of BIND 9.

QryNxrrset This indicates the number of queries that resulted in NOERROR responses with no data. This corresponds to the `nxrrset` counter of previous versions of BIND 9.

QrySERVFAIL This indicates the number of queries that resulted in SERVFAIL.

QryFORMERR This indicates the number of queries that resulted in FORMERR.

QryNXDOMAIN This indicates the number of queries that resulted in NXDOMAIN. This corresponds to the `nxdomain` counter of previous versions of BIND 9.

QryRecursion This indicates the number of queries that caused the server to perform recursion in order to find the final answer. This corresponds to the `recursion` counter of previous versions of BIND 9.

QryDuplicate This indicates the number of queries which the server attempted to recurse but for which it discovered an existing query with the same IP address, port, query ID, name, type, and class already being processed. This corresponds to the `duplicate` counter of previous versions of BIND 9.

QryDropped This indicates the number of recursive queries for which the server discovered an excessive number of existing recursive queries for the same name, type, and class, and which were subsequently dropped. This is the number of dropped queries due to the reason explained with the `clients-per-query` and `max-clients-per-query` options (see *clients-per-query*). This corresponds to the `dropped` counter of previous versions of BIND 9.

QryFailure This indicates the number of query failures. This corresponds to the `failure` counter of previous versions of BIND 9. Note: this counter is provided mainly for backward compatibility with previous versions; normally, more fine-grained counters such as `AuthQryRej` and `RecQryRej` that would also fall into this counter are provided, so this counter is not of much interest in practice.

QryNXRedir This indicates the number of queries that resulted in NXDOMAIN that were redirected.

QryNXRedirRLookup This indicates the number of queries that resulted in NXDOMAIN that were redirected and resulted in a successful remote lookup.

XfrReqDone This indicates the number of requested and completed zone transfers.

UpdateReqFwd This indicates the number of forwarded update requests.

UpdateRespFwd This indicates the number of forwarded update responses.

UpdateFwdFail This indicates the number of forwarded dynamic updates that failed.

UpdateDone This indicates the number of completed dynamic updates.

UpdateFail This indicates the number of failed dynamic updates.

UpdateBadPrereq This indicates the number of dynamic updates rejected due to a prerequisite failure.

RateDropped This indicates the number of responses dropped due to rate limits.

RateSlipped This indicates the number of responses truncated by rate limits.

RPZRewrites This indicates the number of response policy zone rewrites.

Zone Maintenance Statistics Counters

NotifyOutv4 This indicates the number of IPv4 notifies sent.

NotifyOutv6 This indicates the number of IPv6 notifies sent.

NotifyInv4 This indicates the number of IPv4 notifies received.

NotifyInv6 This indicates the number of IPv6 notifies received.

NotifyRej This indicates the number of incoming notifies rejected.

SOAOutv4 This indicates the number of IPv4 SOA queries sent.

SOAOutv6 This indicates the number of IPv6 SOA queries sent.

AXFRReqv4 This indicates the requested IPv4 AXFR.

AXFRReqv6 This indicates the requested IPv6 AXFR.

IXFRReqv4 This indicates the requested IPv4 IXFR.

IXFRReqv6 This indicates the requested IPv6 IXFR.

XfrSuccess This indicates the number of successful zone transfer requests.

XfrFail This indicates the number of failed zone transfer requests.

Resolver Statistics Counters

Queryv4 This indicates the number of IPv4 queries sent.

Queryv6 This indicates the number of IPv6 queries sent.

Responsev4 This indicates the number of IPv4 responses received.

Responsev6 This indicates the number of IPv6 responses received.

NXDOMAIN This indicates the number of NXDOMAINs received.

SERVFAIL This indicates the number of SERVFAILs received.

FORMERR This indicates the number of FORMERRs received.

OtherError This indicates the number of other errors received.

EDNS0Fail This indicates the number of EDNS(0) query failures.

Mismatch This indicates the number of mismatched responses received, meaning 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 `port` option.) This may be an indication of a cache poisoning attempt.

Truncated This indicates the number of truncated responses received.

Lame This indicates the number of lame delegations received.

Retry This indicates the number of query retries performed.

QueryAbort This indicates the number of queries aborted due to quota control.

QuerySockFail This indicates the number of failures in opening query sockets. One common reason for such failures is due to a limitation on file descriptors.

QueryTimeout This indicates the number of query timeouts.

GlueFetchv4 This indicates the number of IPv4 NS address fetches invoked.

GlueFetchv6 This indicates the number of IPv6 NS address fetches invoked.

`GlueFetchv4Fail` This indicates the number of failed IPv4 NS address fetches.

`GlueFetchv6Fail` This indicates the number of failed IPv6 NS address fetches.

`ValAttempt` This indicates the number of attempted DNSSEC validations.

`ValOk` This indicates the number of successful DNSSEC validations.

`ValNegOk` This indicates the number of successful DNSSEC validations on negative information.

`ValFail` This indicates the number of failed DNSSEC validations.

`QryRTTnn` This provides a frequency table on query round-trip times (RTTs). Each `nn` specifies the corresponding frequency. In the sequence of `nn_1`, `nn_2`, ..., `nn_m`, the value of `nn_i` is the number of queries whose RTTs are between `nn_(i-1)` (inclusive) and `nn_i` (exclusive) milliseconds. For the sake of convenience, we define `nn_0` to be 0. The last entry should be represented as `nn_m+`, which means the number of queries whose RTTs are equal to or greater than `nn_m` milliseconds.

Socket I/O Statistics Counters

Socket I/O statistics counters are defined per socket type, which are `UDP4` (UDP/IPv4), `UDP6` (UDP/IPv6), `TCP4` (TCP/IPv4), `TCP6` (TCP/IPv6), `Unix` (Unix Domain), and `FDwatch` (sockets opened outside the socket module). In the following list, `<TYPE>` represents a socket type. Not all counters are available for all socket types; exceptions are noted in the descriptions.

`<TYPE>Open` This indicates the number of sockets opened successfully. This counter does not apply to the `FDwatch` type.

`<TYPE>OpenFail` This indicates the number of failures to open sockets. This counter does not apply to the `FDwatch` type.

`<TYPE>Close` This indicates the number of closed sockets.

`<TYPE>BindFail` This indicates the number of failures to bind sockets.

`<TYPE>ConnFail` This indicates the number of failures to connect sockets.

`<TYPE>Conn` This indicates the number of connections established successfully.

`<TYPE>AcceptFail` This indicates the number of failures to accept incoming connection requests. This counter does not apply to the `UDP` and `FDwatch` types.

`<TYPE>Accept` This indicates the number of incoming connections successfully accepted. This counter does not apply to the `UDP` and `FDwatch` types.

`<TYPE>SendErr` This indicates the number of errors in socket send operations.

`<TYPE>RecvErr` This indicates the number of errors in socket receive operations, including errors of send operations on a connected UDP socket, notified by an ICMP error message.

CHAPTER 5

Advanced DNS Features

5.1 Notify

DNS NOTIFY is a mechanism that allows primary servers to notify their secondary servers of changes to a zone's data. In response to a `NOTIFY` from a primary server, the secondary checks to see that its version of the zone is the current version and, if not, initiates a zone transfer.

For more information about DNS `NOTIFY`, see the description of the `notify` option in *Boolean Options* and the description of the zone option `also-notify` in *Zone Transfers*. The `NOTIFY` protocol is specified in **RFC 1996**.

Note: As a secondary zone can also be a primary to other secondaries, `named`, by default, sends `NOTIFY` messages for every zone it loads. Specifying `notify master-only;` causes `named` to only send `NOTIFY` for primary zones that it loads.

5.2 Dynamic Update

Dynamic update is a method for adding, replacing, or deleting records in a primary server by sending it a special form of DNS messages. The format and meaning of these messages is specified in **RFC 2136**.

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 are permitted for the key `local-ddns`, which is generated by `named` at startup. See *Dynamic Update Policies* for more details.

Dynamic updates using Kerberos-signed requests can be made using the TKEY/GSS protocol, either by setting the `tkey-gssapi-keytab` option or by setting both the `tkey-gssapi-credential` and `tkey-domain` options. Once enabled, Kerberos-signed requests are 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.

5.2.1 The Journal File

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 also occasionally writes ("dumps") 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 are created with the extensions `.jnw` and `.jbk`; under ordinary circumstances, these are removed when the dump is complete, and can be safely ignored.

When a server is restarted after a shutdown or crash, it replays 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 journaled 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`.

To make changes to a dynamic zone manually, follow these steps: First, disable dynamic updates to the zone using `rndc freeze zone`; this updates the zone's master file with the changes stored in its `.jnl` file. Then, edit the zone file. Finally, run `rndc thaw zone` to reload the changed zone and re-enable dynamic updates.

`rndc sync zone` updates 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`.

5.3 Incremental Zone Transfers (IXFR)

The incremental zone transfer (IXFR) protocol is a way for secondary servers to transfer only changed data, instead of having to transfer an entire zone. The IXFR protocol is specified in **RFC 1995**. See *Proposed Standards*.

When acting as a primary server, BIND 9 supports IXFR for those zones where the necessary change history information is available. These include primary zones maintained by dynamic update and secondary zones whose data was obtained by IXFR. For manually maintained primary zones, and for secondary 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 secondary server, BIND 9 attempts 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.

When a secondary server receives a zone via AXFR, it creates a new copy of the zone database and then swaps it into place; during the loading process, queries continue to be served from the old database with no interference. When receiving a zone via IXFR, however, changes are applied to the running zone, which may degrade query performance during the transfer. If a server receiving an IXFR request determines that the response size would be similar in size to an AXFR response, it may wish to send AXFR instead. The threshold at which this determination is made can be configured using the `max-ixfr-ratio` option.

5.4 Split DNS

Setting up different views 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 to use Split DNS is to hide "internal" DNS information from "external" clients on the Internet. There is some debate as to whether 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 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 into the internal network.

5.4.1 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.

To accomplish this, the company sets up two sets of name servers. One set is on the inside network (in the reserved IP space) and the other set is on bastion hosts, which are "proxy" hosts in the DMZ that can talk to both sides of its network.

The internal servers are 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 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, are configured to serve the "public" version of the `site1.example.com` 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.example.com` 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 have no other way of determining how to deliver mail to those internal hosts. With the wildcard records, the mail is 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 need to know how to deliver mail to internal hosts. The resolvers on the bastion hosts need to be configured to point to the internal name servers for DNS resolution.

Queries for internal hostnames are answered by the internal servers, and queries for external hostnames are forwarded back out to the DNS servers on the bastion hosts.

For all of this to work properly, internal clients 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 are now able to:

- Look up any hostnames in the `site1.example.com` 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 users.

Hosts on the Internet are able to:

- Look up any hostnames in the `site1.example.com` and `site2.example.com` zones.
- Exchange mail with anyone in the `site1.example.com` and `site2.example.com` zones.

Here is an example configuration for the setup just described above. Note that this is only configuration information; for information on how to configure the zone files, see *Sample Configurations*.

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 primary zone
zone "site1.example.com" {
  type master;
  file "m/site1.example.com";
  // do normal iterative resolution (do not forward)
  forwarders { };
  allow-query { internals; externals; };
  allow-transfer { internals; };
};

// sample secondary zone
zone "site2.example.com" {
```

(continues on next page)

(continued from previous page)

```
  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 configuration:

```
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; };
  // restrict cache access
  allow-query-cache { internals; externals; };
  // restrict recursion
  allow-recursion { internals; externals; };
  ...
  ...
};

// sample secondary zone
zone "site1.example.com" {
  type master;
  file "m/site1.foo.com";
  allow-transfer { internals; externals; };
};
```

(continues on next page)

(continued from previous page)

```
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
```

5.5 TSIG

TSIG (Transaction SIGnatures) is a mechanism for authenticating DNS messages, originally specified in **RFC 2845**. It allows DNS messages to be cryptographically signed using a shared secret. TSIG can be used in any DNS transaction, as a way to restrict access to certain server functions (e.g., recursive queries) 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 the server, such as with dynamic UPDATE messages or zone transfers from a primary to a secondary server.

This section is a guide to setting up TSIG in BIND. 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 BIND support it for sending messages to `named`:

- *nsupdate - dynamic DNS update utility* supports TSIG via the `-k`, `-l` and `-y` command line options, or via the `key` command when running interactively.
- *dig - DNS lookup utility* supports TSIG via the `-k` and `-y` command line options.

5.5.1 Generating a Shared Key

TSIG keys can be generated using the `tsig-keygen` command; 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 "tsig-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:

```
$ tsig-keygen 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. (Note: copying a shared secret from one server to another is beyond the scope of the DNS. A secure transport mechanism should be used: secure FTP, SSL, ssh, telephone, encrypted email, etc.)

`tsig-keygen` can also be run as `ddns-confgen`, in which case its output includes additional configuration text for setting up dynamic DNS in `named`. See *ddns-confgen - ddns key generation tool* for details.

5.5.2 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 `tsig-keygen`.)

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 reconfigured, the server can recognize the key. If the server receives a message signed by the key, it is able to verify the signature. If the signature is valid, the response is signed using the same key.

TSIG keys that are known to a server can be listed using the command `rndc tsig-list`.

5.5.3 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 secondary zone; in this case, all SOA QUERY messages, NOTIFY messages, and zone transfer requests (AXFR or IXFR) are signed using the specified key. Keys may also be specified in the `also-notify` statement of a primary or secondary 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.

When any server sends a TSIG-signed DNS request, it expects the response to be signed with the same key. If a response is not signed, or if the signature is not valid, the response is rejected.

5.5.4 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.`

Here's 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 *Dynamic Update Policies* for a discussion of the more flexible `update-policy` statement.

5.5.5 Errors

Processing of TSIG-signed messages can result in several 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 but 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 returns a response code of NOTAUTH (not authenticated).

5.6 TKEY

TKEY (Transaction KEY) is a mechanism for automatically negotiating a shared secret between two hosts, originally specified in **RFC 2930**.

There are several TKEY "modes" that specify how a key is to be generated or assigned. BIND 9 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, contains a TKEY record in its answer section. After this transaction, both participants have enough information to calculate a shared secret using Diffie-Hellman key exchange. The shared secret can then be used 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.

5.7 SIG(0)

BIND partially supports DNSSEC 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 is only verified if the key is known and trusted by the server. The server does not attempt to recursively fetch or validate the key.

SIG(0) signing of multiple-message TCP streams is not supported.

The only tool shipped with BIND 9 that generates SIG(0) signed messages is `nsupdate`.

5.8 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. BIND 9 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, and that the tools shipped with BIND 9.2.x and earlier are not compatible with the current versions.

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.

5.8.1 Generating Keys

The `dnssec-keygen` program is used to generate keys.

A secure zone must contain one or more zone keys. The zone keys 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, have a name type of `ZONE`, and be usable for authentication. It is recommended that zone keys use a cryptographic algorithm designated as "mandatory to implement" by the IETF. Currently there are two algorithms, RSASHA256 and ECDSAP256SHA256; ECDSAP256SHA256 is recommended for current and future deployments.

The following command generates a ECDSAP256SHA256 key for the `child.example` zone:

```
dnssec-keygen -a ECDSAP256SHA256 -n ZONE child.example.
```

Two output files are produced: `Kchild.example.+013+12345.key` and `Kchild.example.+013+12345.private` (where 12345 is an example of a key tag). The key filenames contain the key name (`child.example.`), the algorithm (5 is RSASHA1, 8 is RSASHA256, 13 is ECDSAP256SHA256, 15 is ED25519, 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` program is used to get a key pair from a crypto hardware and build the key files. Its usage is similar to `dnssec-keygen`.

The public keys should be inserted into the zone file by including the `.key` files using `$INCLUDE` statements.

5.8.2 Signing the Zone

The `dnssec-signzone` program is used to sign a zone.

Any `keyset` files corresponding to secure sub-zones should be present. The zone signer generates `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.

By default, all zone keys which have an available private key are used to generate signatures. The following command signs the zone, assuming it is in a file called `zone.child.example`:

```
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` also produces a keyset and dsset files. 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.

5.8.3 Configuring Servers for DNSSEC

To enable `named` to validate answers received from other servers, the `dnssec-validation` option must be set to either `yes` or `auto`.

When `dnssec-validation` is set to `auto`, a trust anchor for the DNS root zone is automatically used. This trust anchor is provided as part of BIND and is kept up to date using **RFC 5011** key management.

When `dnssec-validation` is set to `yes`, DNSSEC validation only occurs if at least one trust anchor has been explicitly configured in `named.conf`, using a `trust-anchors` statement (or the `managed-keys` and `trusted-keys` statements, both deprecated).

When `dnssec-validation` is set to `no`, DNSSEC validation does not occur.

The default is `auto` unless BIND is built with `configure --disable-auto-validation`, in which case the default is `yes`.

The keys specified in `trust-anchors` are copies of DNSKEY RRs for zones that are used to form the first link in the cryptographic chain of trust. Keys configured with the keyword `static-key` or `static-ds` are loaded directly into the table of trust anchors, and can only be changed by altering the configuration. Keys configured with `initial-key` or `initial-ds` are used to initialize **RFC 5011** trust anchor maintenance, and are kept up-to-date automatically after the first time `named` runs.

`trust-anchors` is described in more detail later in this document.

BIND 9 does not verify signatures on load, so zone keys for authoritative zones do not need to be specified in the configuration file.

After DNSSEC is established, a typical DNSSEC configuration looks something like the following. It has one or more public keys for the root, which allows answers from outside the organization to be validated. It also has several keys for parts of the namespace that the organization controls. These are here to ensure that `named` is immune to compromised security in the DNSSEC components of parent zones.

```
trust-anchors {
    /* Root Key */
    "." initial-key 257 3 3 "BNY4wrWM1nCfJ+CXd0rVXyYmobt7sEEfK3clRbGaTwS
                JxrGkxJWoZu6I7PzJu/E9gx4UC1zGAHlXKdE4zYIpRh
                aBKnvcC2U9mZhkdUpd1Vso/HAdjNe8LmMlnzY3zy2Xy
                4klWOADTPzSv9eamj8V18PHGjBLaVtYvk/ln5ZApjYg
                hf+6fElrmLkdaz MQ2OCnACR817DF4BBa7UR/beDHyp
                5iWTXWSi6XmoJLbG9Scqc7l70KDqlvXR3M/lUUVRbke
                g1IPJSidmK3ZyCllh4XSKbje/45SKucHgnwU5jefMtq
                66gKodQj+MiA21AfUVe7u99WzTLzY3qlxDhxYQQ2OFQ
                97S+LKUTpQcq27R7AT3/V5hRQxScINqwcz4jYqZD2fQ
                dgxbcDTClUOCRBdiieyLMNzXG3";
    /* Key for our organization's forward zone */
    example.com. static-ds 54135 5 2
↪"8EF922C97F1D07B23134440F19682E7519ADDAE180E20B1B1EC52E7F58B2831D"
```

(continues on next page)

(continued from previous page)

```
    /* Key for our reverse zone. */
    2.0.192.IN-ADDRPA.NET. static-key 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
                        DIRvhDD52SKvbheeTJUm6Ehkz
                        ytNN2SN96QRk8j/iI8ib";
};

options {
    ...
    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 rejects any answers from signed, secure zones which fail to validate, and returns SERVFAIL 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, the validator 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."

5.9 DNSSEC, Dynamic Zones, and Automatic Signing

5.9.1 Converting From Insecure to Secure

Changing a zone from insecure to secure can be done in two ways: using a dynamic DNS update, or via the `auto-dnssec` zone option.

For either method, `named` must be configured so that it can see the `K*` files which contain the public and private parts of the keys that are used to sign the zone. These files are generated by `dnssec-keygen`, and they should be placed 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 causes all records in the zone to be signed with the ZSK, and the DNSKEY RRset to be signed with the KSK. An NSEC chain is generated as part of the initial signing process.

5.9.2 Dynamic DNS Update Method

To insert the keys via dynamic update:

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7␣
↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8al35zz3Y I1m/
↪SAQBxIqMfLtIwqWPdgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+X3e/
↪UNlq9IHq3Y0 XtC0Iuawl/qkaKVxXe2lo8Ct+dM6UehyCqk=
> send
```

While the update request completes almost immediately, the zone is not completely signed until `named` has had time to walk the zone and generate the NSEC and RRSIG records. The NSEC record at the apex is added last, to signal that there is a complete NSEC chain.

To sign using NSEC3 instead of NSEC, add an NSEC3PARAM record to the initial update request. The OPTOUT bit in the NSEC3 chain can be set in the flags field of the NSEC3PARAM record.

```
% nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7␣
↪AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8al35zz3Y I1m/
↪SAQBxIqMfLtIwqWPdgthsu36azGQAX8=
> update add example.net DNSKEY 257 3 7 AwEAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+X3e/
↪UNlq9IHq3Y0 XtC0Iuawl/qkaKVxXe2lo8Ct+dM6UehyCqk=
> update add example.net NSEC3PARAM 1 1 100 1234567890
> send
```

Again, this update request completes almost immediately; however, the record does not show up until `named` has had a chance to build/remove the relevant chain. A private type record is created to record the state of the operation (see below for more details), and is removed once the operation completes.

While the initial signing and NSEC/NSEC3 chain generation is happening, other updates are possible as well.

5.9.3 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 does so only when it receives an `rndc sign <zonename>`.

`auto-dnssec maintain` includes the above functionality, but also automatically adjusts the zone's DNSKEY records on a schedule according to the keys' timing metadata. (See *dnssec-keygen: DNSSEC key generation tool* and *dnssec-settime: set the key timing metadata for a DNSSEC key* for more information.)

`named` periodically searches the key directory for keys matching the zone; if the keys' metadata indicates that any change should be made to the zone, such as adding, removing, or revoking a key, then that action is 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` command 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 is signed immediately, without waiting for an `rndc sign` or `rndc loadkeys` command. Those commands can still be used when there are unscheduled key changes.

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, the TTL is set to the TTL specified when the key was created (using the `dnssec-keygen -L` option), if any, or to the SOA TTL.

To sign the zone using NSEC3 instead of NSEC, submit an NSEC3PARAM record via dynamic update prior to the scheduled publication and activation of the keys. The OPTOUT bit for the NSEC3 chain can be set in the flags field of the NSEC3PARAM record. The NSEC3PARAM record does not appear in the zone immediately, but it is stored for later reference. When the zone is signed and the NSEC3 chain is completed, the NSEC3PARAM record appears 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 fails.

5.9.4 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, those records with a nonzero initial octet have a nonzero value for the final octet.

If the first octet of a private-type record is non-zero, the record indicates either 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 are silently ignored.

If the first octet is zero (this is a reserved algorithm number that should never appear in a DNSKEY record), the record indicates changes to the NSEC3 chains are in progress. The rest of the record contains an NSEC3PARAM record, while the flag field tells what operation to perform based on the flag bits.

0x01 OPTOUT

0x80 CREATE

0x40 REMOVE

0x20 NONSEC

5.9.5 DNSKEY Rollovers

As with insecure-to-secure conversions, DNSSEC keyrolls can be done in two ways: using a dynamic DNS update, or via the `auto-dnssec` zone option.

5.9.6 Dynamic DNS Update Method

To perform key rollovers via dynamic update, the `K*` files for the new keys must be added so that `named` can find them. The new DNSKEY RRs can then be added via dynamic update. `named` then causes the zone to be signed with the new keys; when the signing is complete, the private-type records are updated so that the last octet is non-zero.

If this is for a KSK, the parent and any trust anchor repositories of the new KSK must be informed.

The maximum TTL in the zone must expire before removing the old DNSKEY. If it is a KSK that is being updated, the DS RRset in the parent must also be updated and its TTL allowed to expire. This ensures that all clients are able to verify at least one signature when the old DNSKEY is removed.

The old DNSKEY can be removed via UPDATE, taking care to specify the correct key. `named` cleans out any signatures generated by the old key after the update completes.

5.9.7 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` automatically carries out the key rollover. If the key's algorithm has not previously been used to sign the zone, then the zone is fully signed as quickly as possible. However, if the new key replaces an existing key of the same algorithm, the zone is re-signed incrementally, with signatures from the old key replaced with signatures from the new key as their signature validity periods expire. By default, this rollover completes in 30 days, after which it is safe to remove the old key from the DNSKEY RRset.

5.9.8 NSEC3PARAM Rollovers via UPDATE

The new NSEC3PARAM record can be added via dynamic update. When the new NSEC3 chain has been generated, the NSEC3PARAM flag field is set to zero. At that point, the old NSEC3PARAM record can be removed. The old chain is removed after the update request completes.

5.9.9 Converting From NSEC to NSEC3

To do this, an NSEC3PARAM record must be added. When the conversion is complete, the NSEC chain is removed and the NSEC3PARAM record has a zero flag field. The NSEC3 chain is generated before the NSEC chain is destroyed.

5.9.10 Converting From NSEC3 to NSEC

To do this, use `nsupdate` to remove all NSEC3PARAM records with a zero flag field. The NSEC chain is generated before the NSEC3 chain is removed.

5.9.11 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 are removed automatically. This takes 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; otherwise it will re-sign.

5.9.12 Periodic Re-signing

In any secure zone which supports dynamic updates, `named` periodically re-signs RRsets which have not been re-signed as a result of some update action. The signature lifetimes are adjusted to spread the re-sign load over time rather than all at once.

5.9.13 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; if the OPTOUT state of an individual NSEC3 needs to be changed, the entire chain must be changed.

5.10 Dynamic Trust Anchor Management

BIND 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.

5.10.1 Validating Resolver

To configure a validating resolver to use **RFC 5011** to maintain a trust anchor, configure the trust anchor using a `dnssec-keys` statement and the `initial-key` keyword. Information about this can be found in *dnssec-keys Statement Definition and Usage*.

5.10.2 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 KSKs 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 takes note of the stand-by KSKs in the zone's DNSKEY RRset, and stores them for future reference. The

resolver rechecks the zone periodically; after 30 days, if the new key is still there, the key is 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 exists with a publication date in the past, but an activation date which is unset or in the future, " `dnssec-signzone -S`" includes the DNSKEY record in the zone but does 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 command `dnssec-revoke` has been added. This adds the REVOKED bit 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 is never again accepted as a valid trust anchor by the resolver. However, validation can proceed using the new active key, which was 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 IDs exactly 128 apart and one is revoked, the two key IDs will collide, causing several problems. To prevent this, `dnssec-keygen` does not generate a new key if another key is present which may collide. This checking only occurs if the new keys are written to the same directory which holds all other keys in use for that zone.

Older versions of BIND 9 did not have this precaution. Exercise caution if using key revocation on keys that were generated by previous releases, or if using keys stored in multiple directories or on multiple machines.

It is expected that a future release of BIND 9 will address this problem in a different way, by storing revoked keys with their original unrevoked key IDs.

5.11 PKCS#11 (Cryptoki) support

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.

BIND 9 is known to work with three HSMs: The AEP Keyper, which has been tested with Debian Linux, Solaris x86 and Windows Server 2003; the Thales nShield, tested with Debian Linux; and the Sun SCA 6000 cryptographic acceleration board, tested with Solaris x86. In addition, BIND can be used with all current versions of SoftHSM, a software-based HSM simulator library produced by the OpenDNSSEC project.

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.

There are two available mechanisms for PKCS#11 support in BIND 9: OpenSSL-based PKCS#11 and native PKCS#11. When using the first mechanism, BIND uses a modified version of OpenSSL, which loads the provider library and operates the HSM indirectly; any cryptographic operations not supported by the HSM can be carried out by OpenSSL instead. The second mechanism enables BIND to bypass OpenSSL completely; BIND loads the provider library itself, and uses the PKCS#11 API to drive the HSM directly.

5.11.1 Prerequisites

See the documentation provided by your HSM vendor for information about installing, initializing, testing and troubleshooting the HSM.

5.11.2 Native PKCS#11

Native PKCS#11 mode will only work with an HSM capable of carrying out *every* cryptographic operation BIND 9 may need. The HSM's provider library must have a complete implementation of the PKCS#11 API, so that all these functions are accessible. As of this writing, only the Thales nShield HSM and SoftHSMv2 can be used in this fashion. For other HSMs, including the AEP Keyper, Sun SCA 6000 and older versions of SoftHSM, use OpenSSL-based PKCS#11. (Note: Eventually, when more HSMs become capable of supporting native PKCS#11, it is expected that OpenSSL-based PKCS#11 will be deprecated.)

To build BIND with native PKCS#11, configure as follows:

```
$ cd bind9
$ ./configure --enable-native-pkcs11 \
    --with-pkcs11=provider-library-path
```

This will cause all BIND tools, including `named` and the `dnssec-*` and `pkcs11-*` tools, to use the PKCS#11 provider library specified in provider-library-path for cryptography. (The provider library path can be overridden using the `-E` in `named` and the `dnssec-*` tools, or the `-m` in the `pkcs11-*` tools.)

Building SoftHSMv2

SoftHSMv2, the latest development version of SoftHSM, is available from https://github.com/opendnssec/SoftHSMv2. It is a software library developed by the OpenDNSSEC project (http://www.opendnssec.org) which provides a PKCS#11 interface to a virtual HSM, implemented in the form of a SQLite3 database on the local filesystem. It provides less security than a true HSM, but it allows you to experiment with native PKCS#11 when an HSM is not available. SoftHSMv2 can be configured to use either OpenSSL or the Botan library to perform cryptographic functions, but when using it for native PKCS#11 in BIND, OpenSSL is required.

By default, the SoftHSMv2 configuration file is prefix/etc/softhsm2.conf (where prefix is configured at compile time). This location can be overridden by the SOFTHSM2_CONF environment variable. The SoftHSMv2 cryptographic store must be installed and initialized before using it with BIND.

```
$  cd SoftHSMv2
$  configure --with-crypto-backend=openssl --prefix=/opt/pkcs11/usr
$  make
$  make install
$  /opt/pkcs11/usr/bin/softhsm-util --init-token 0 --slot 0 --label softhsmv2
```

5.11.3 OpenSSL-based PKCS#11

OpenSSL-based PKCS#11 mode uses a modified version of the OpenSSL library; stock OpenSSL does not fully support PKCS#11. ISC provides a patch to OpenSSL to correct this. This patch is based on work originally done by the OpenSolaris project; it has been modified by ISC to provide new features such as PIN management and key-by-reference.

There are two "flavors" of PKCS#11 support provided by the patched OpenSSL, one of which must be chosen at configuration time. The correct choice depends on the HSM hardware:

- Use 'crypto-accelerator' with HSMs that have hardware cryptographic acceleration features, such as the SCA 6000 board. This causes OpenSSL to run all supported cryptographic operations in the HSM.
- Use 'sign-only' with HSMs that are designed to function primarily as secure key storage devices, but lack hardware acceleration. These devices are highly secure, but are not necessarily any faster at cryptography than the system CPU MDASH often, they are slower. It is therefore most efficient to use them only for those cryptographic functions that require access to the secured private key, such as zone signing, and to use the system CPU for all other computationally-intensive operations. The AEP Keyper is an example of such a device.

The modified OpenSSL code is included in the BIND 9 release, in the form of a context diff against the latest versions of OpenSSL. OpenSSL 0.9.8, 1.0.0, 1.0.1 and 1.0.2 are supported; there are separate diffs for each version. In the examples to follow, we use OpenSSL 0.9.8, but the same methods work with OpenSSL 1.0.0 through 1.0.2.

Note: The OpenSSL patches as of this writing (January 2016) support versions 0.9.8zh, 1.0.0t, 1.0.1q and 1.0.2f. ISC will provide updated patches as new versions of OpenSSL are released. The version number in the following examples is expected to change.

Before building BIND 9 with PKCS#11 support, it will be necessary to build OpenSSL with the patch in place, and configure it with the path to your HSM's PKCS#11 provider library.

Patching OpenSSL

```
$ wget http://www.openssl.org/source/openssl-0.9.8zc.tar.gz
```

Extract the tarball:

```
$ tar zxf openssl-0.9.8zc.tar.gz
```

Apply the patch from the BIND 9 release:

```
$ patch -p1 -d openssl-0.9.8zc \
          < bind9/bin/pkcs11/openssl-0.9.8zc-patch
```

Note: The patch file may not be compatible with the "patch" utility on all operating systems. You may need to install GNU patch.

When building OpenSSL, place it in a non-standard location so that it does not interfere with OpenSSL libraries elsewhere on the system. In the following examples, we choose to install into "/opt/pkcs11/usr". We will use this location when we configure BIND 9.

Later, when building BIND 9, the location of the custom-built OpenSSL library will need to be specified via configure.

Building OpenSSL for the AEP Keyper on Linux

The AEP Keyper is a highly secure key storage device, but does not provide hardware cryptographic acceleration. It can carry out cryptographic operations, but it is probably slower than your system's CPU. Therefore, we choose the 'sign-only' flavor when building OpenSSL.

The Keyper-specific PKCS#11 provider library is delivered with the Keyper software. In this example, we place it /opt/pkcs11/usr/lib:

```
$ cp pkcs11.GCC4.0.2.so.4.05 /opt/pkcs11/usr/lib/libpkcs11.so
```

```
$ cd openssl-0.9.8zc
$ ./Configure linux-x86_64 \
        --pk11-libname=/opt/pkcs11/usr/lib/libpkcs11.so \
        --pk11-flavor=sign-only \
        --prefix=/opt/pkcs11/usr
```

Building OpenSSL for the SCA 6000 on Solaris

The SCA-6000 PKCS#11 provider is installed as a system library, libpkcs11. It is a true crypto accelerator, up to 4 times faster than any CPU, so the flavor shall be 'crypto-accelerator'.

In this example, we are building on Solaris x86 on an AMD64 system.

```
$ cd openssl-0.9.8zc
$ ./Configure solaris64-x86_64-cc \
        --pk11-libname=/usr/lib/64/libpkcs11.so \
        --pk11-flavor=crypto-accelerator \
        --prefix=/opt/pkcs11/usr
```

(For a 32-bit build, use "solaris-x86-cc" and /usr/lib/libpkcs11.so.)

After configuring, run `make` and `make test`.

Building OpenSSL for SoftHSM

SoftHSM (version 1) is a software library developed by the OpenDNSSEC project (http://www.opendnssec.org) which provides a PKCS#11 interface to a virtual HSM, implemented in the form of a SQLite3 database on the local filesystem. SoftHSM uses the Botan library to perform cryptographic functions. Though less secure than a true HSM, it can allow you to experiment with PKCS#11 when an HSM is not available.

The SoftHSM cryptographic store must be installed and initialized before using it with OpenSSL, and the SOFTHSM_CONF environment variable must always point to the SoftHSM configuration file:

```
$  cd softhsm-1.3.7
$  configure --prefix=/opt/pkcs11/usr
$  make
$  make install
$  export SOFTHSM_CONF=/opt/pkcs11/softhsm.conf
$  echo "0:/opt/pkcs11/softhsm.db" > $SOFTHSM_CONF
$  /opt/pkcs11/usr/bin/softhsm --init-token 0 --slot 0 --label softhsm
```

SoftHSM can perform all cryptographic operations, but since it only uses your system CPU, there is no advantage to using it for anything but signing. Therefore, we choose the 'sign-only' flavor when building OpenSSL.

```
$ cd openssl-0.9.8zc
$ ./Configure linux-x86_64 \
        --pk11-libname=/opt/pkcs11/usr/lib/libsofthsm.so \
```

(continues on next page)

(continued from previous page)

```
        --pk11-flavor=sign-only \
        --prefix=/opt/pkcs11/usr
```

After configuring, run "`make`" and "`make test`".

Once you have built OpenSSL, run "`apps/openssl engine pkcs11`" to confirm that PKCS#11 support was compiled in correctly. The output should be one of the following lines, depending on the flavor selected:

```
(pkcs11) PKCS #11 engine support (sign only)
```

Or:

```
(pkcs11) PKCS #11 engine support (crypto accelerator)
```

Next, run "`apps/openssl engine pkcs11 -t`". This will attempt to initialize the PKCS#11 engine. If it is able to do so successfully, it will report "`[ available ]`".

If the output is correct, run "`make install`" which will install the modified OpenSSL suite to `/opt/pkcs11/usr`.

Configuring BIND 9 for Linux with the AEP Keyper

```
$ cd ../bind9
$ ./configure \
       --with-openssl=/opt/pkcs11/usr \
       --with-pkcs11=/opt/pkcs11/usr/lib/libpkcs11.so
```

Configuring BIND 9 for Solaris with the SCA 6000

```
$ cd ../bind9
$ ./configure CC="cc -xarch=amd64" \
        --with-openssl=/opt/pkcs11/usr \
        --with-pkcs11=/usr/lib/64/libpkcs11.so
```

(For a 32-bit build, omit CC="cc -xarch=amd64".)

If configure complains about OpenSSL not working, you may have a 32/64-bit architecture mismatch. Or, you may have incorrectly specified the path to OpenSSL (it should be the same as the –prefix argument to the OpenSSL Configure).

Configuring BIND 9 for SoftHSM

```
$ cd ../bind9
$ ./configure \
       --with-openssl=/opt/pkcs11/usr \
       --with-pkcs11=/opt/pkcs11/usr/lib/libsofthsm.so
```

After configuring, run "`make`", "`make test`" and "`make install`".

(Note: If "make test" fails in the "pkcs11" system test, you may have forgotten to set the SOFTHSM_CONF environment variable.)

5.11.4 PKCS#11 Tools

BIND 9 includes a minimal set of tools to operate the HSM, including `pkcs11-keygen` to generate a new key pair within the HSM, `pkcs11-list` to list objects currently available, `pkcs11-destroy` to remove objects, and `pkcs11-tokens` to list available tokens.

In UNIX/Linux builds, these tools are built only if BIND 9 is configured with the –with-pkcs11 option. (Note: If –with-pkcs11 is set to "yes", rather than to the path of the PKCS#11 provider, then the tools will be built but the provider will be left undefined. Use the -m option or the PKCS11_PROVIDER environment variable to specify the path to the provider.)

5.11.5 Using the HSM

For OpenSSL-based PKCS#11, we must first set up the runtime environment so the OpenSSL and PKCS#11 libraries can be loaded:

```
$ export LD_LIBRARY_PATH=/opt/pkcs11/usr/lib:${LD_LIBRARY_PATH}
```

This causes `named` and other binaries to load the OpenSSL library from `/opt/pkcs11/usr/lib` rather than from the default location. This step is not necessary when using native PKCS#11.

Some HSMs require other environment variables to be set. For example, when operating an AEP Keyper, it is necessary to specify the location of the "machine" file, which stores information about the Keyper for use by the provider library. If the machine file is in `/opt/Keyper/PKCS11Provider/machine`, use:

```
$ export KEYPER_LIBRARY_PATH=/opt/Keyper/PKCS11Provider
```

Such environment variables must be set whenever running any tool that uses the HSM, including `pkcs11-keygen`, `pkcs11-list`, `pkcs11-destroy`, `dnssec-keyfromlabel`, `dnssec-signzone`, `dnssec-keygen`, and `named`.

We can now create and use keys in the HSM. In this case, we will create a 2048 bit key and give it the label "sample-ksk":

```
$ pkcs11-keygen -b 2048 -l sample-ksk
```

To confirm that the key exists:

```
$ pkcs11-list
Enter PIN:
object[0]: handle 2147483658 class 3 label[8] 'sample-ksk' id[0]
object[1]: handle 2147483657 class 2 label[8] 'sample-ksk' id[0]
```

Before using this key to sign a zone, we must create a pair of BIND 9 key files. The "dnssec-keyfromlabel" utility does this. In this case, we will be using the HSM key "sample-ksk" as the key-signing key for "example.net":

```
$ dnssec-keyfromlabel -l sample-ksk -f KSK example.net
```

The resulting K*.key and K*.private files can now be used to sign the zone. Unlike normal K* files, which contain both public and private key data, these files will contain only the public key data, plus an identifier for the private key which remains stored within the HSM. Signing with the private key takes place inside the HSM.

If you wish to generate a second key in the HSM for use as a zone-signing key, follow the same procedure above, using a different keylabel, a smaller key size, and omitting "-f KSK" from the dnssec-keyfromlabel arguments:

(Note: When using OpenSSL-based PKCS#11 the label is an arbitrary string which identifies the key. With native PKCS#11, the label is a PKCS#11 URI string which may include other details about the key and the HSM, including its PIN. See *dnssec-keyfromlabel - DNSSEC key generation tool* for details.)

```
$ pkcs11-keygen -b 1024 -l sample-zsk
$ dnssec-keyfromlabel -l sample-zsk example.net
```

Alternatively, you may prefer to generate a conventional on-disk key, using dnssec-keygen:

```
$ dnssec-keygen example.net
```

This provides less security than an HSM key, but since HSMs can be slow or cumbersome to use for security reasons, it may be more efficient to reserve HSM keys for use in the less frequent key-signing operation. The zone-signing key can be rolled more frequently, if you wish, to compensate for a reduction in key security. (Note: When using native PKCS#11, there is no speed advantage to using on-disk keys, as cryptographic operations will be done by the HSM regardless.)

Now you can sign the zone. (Note: If not using the -S option to `dnssec-signzone`, it will be necessary to add the contents of both `K*.key` files to the zone master file before signing it.)

```
$ dnssec-signzone -S example.net
Enter PIN:
Verifying the zone using the following algorithms:
NSEC3RSASHA1.
Zone signing complete:
Algorithm: NSEC3RSASHA1: ZSKs: 1, KSKs: 1 active, 0 revoked, 0 stand-by
example.net.signed
```

5.11.6 Specifying the engine on the command line

When using OpenSSL-based PKCS#11, the "engine" to be used by OpenSSL can be specified in `named` and all of the BIND `dnssec-*` tools by using the "-E <engine>" command line option. If BIND 9 is built with the –with-pkcs11 option, this option defaults to "pkcs11". Specifying the engine will generally not be necessary unless for some reason you wish to use a different OpenSSL engine.

If you wish to disable use of the "pkcs11" engine MDASH for troubleshooting purposes, or because the HSM is unavailable MDASH set the engine to the empty string. For example:

```
$ dnssec-signzone -E '' -S example.net
```

This causes `dnssec-signzone` to run as if it were compiled without the –with-pkcs11 option.

When built with native PKCS#11 mode, the "engine" option has a different meaning: it specifies the path to the PKCS#11 provider library. This may be useful when testing a new provider library.

5.11.7 Running named with automatic zone re-signing

If you want `named` to dynamically re-sign zones using HSM keys, and/or to to sign new records inserted via nsupdate, then `named` must have access to the HSM PIN. In OpenSSL-based PKCS#11, this is accomplished by placing the PIN into the openssl.cnf file (in the above examples, `/opt/pkcs11/usr/ssl/openssl.cnf`).

The location of the openssl.cnf file can be overridden by setting the OPENSSL_CONF environment variable before running `named`.

Sample openssl.cnf:

```
openssl_conf = openssl_def
[ openssl_def ]
engines = engine_section
[ engine_section ]
pkcs11 = pkcs11_section
[ pkcs11_section ]
PIN = <PLACE PIN HERE>
```

This will also allow the dnssec-* tools to access the HSM without PIN entry. (The pkcs11-* tools access the HSM directly, not via OpenSSL, so a PIN will still be required to use them.)

In native PKCS#11 mode, the PIN can be provided in a file specified as an attribute of the key's label. For example, if a key had the label `pkcs11:object=local-zsk;pin-source=/etc/hsmpin`, then the PIN would be read from the file `/etc/hsmpin`.

Warning: Placing the HSM's PIN in a text file in this manner may reduce the security advantage of using an HSM. Be sure this is what you want to do before configuring the system in this way.

5.12 Dynamically Loadable Zones (DLZ)

Dynamically Loadable Zones (DLZ) are an extension to BIND 9 that allows zone data to be retrieved directly from an external database. There is no required format or schema. DLZ drivers exist for several different database backends, including PostgreSQL, MySQL, and LDAP, and can be written for any other.

Historically, DLZ drivers had to be statically linked with the `named` binary and were turned on via a configure option at compile time (for example, `configure --with-dlz-ldap`). The drivers provided in the BIND 9 tarball in `contrib/dlz/drivers` are still linked this way.

In BIND 9.8 and higher, it is possible to link some DLZ modules dynamically at runtime, via the DLZ "dlopen" driver, which acts as a generic wrapper around a shared object implementing the DLZ API. The "dlopen" driver is linked into `named` by default, so configure options are no longer necessary when using these dynamically linkable drivers, but are still needed for the older drivers in `contrib/dlz/drivers`.

The DLZ module provides data to `named` in text format, which is then converted to DNS wire format by `named`. This conversion, and the lack of any internal caching, places significant limits on the query performance of DLZ modules. Consequently, DLZ is not recommended for use on high-volume servers. However, it can be used in a hidden primary (master) configuration, with secondaries retrieving zone updates via AXFR. Note, however, that DLZ has no built-in support for DNS notify; secondary servers are not automatically informed of changes to the zones in the database.

5.12.1 Configuring DLZ

A DLZ database is configured with a `dlz` statement in `named.conf`:

```
dlz example {
database "dlopen driver.so args";
```

(continues on next page)

(continued from previous page)

```
search yes;
};
```

This specifies a DLZ module to search when answering queries; the module is implemented in `driver.so` and is loaded at runtime by the dlopen DLZ driver. Multiple `dlz` statements can be specified; when answering a query, all DLZ modules with `search` set to `yes` are queried to see whether they contain an answer for the query name. The best available answer is returned to the client.

The `search` option in the above example can be omitted, because `yes` is the default value.

If `search` is set to `no`, this DLZ module is *not* searched for the best match when a query is received. Instead, zones in this DLZ must be separately specified in a zone statement. This allows users to configure a zone normally using standard zone-option semantics, but specify a different database backend for storage of the zone's data. For example, to implement NXDOMAIN redirection using a DLZ module for backend storage of redirection rules:

```
dlz other {
database "dlopen driver.so args";
search no;
};

zone "." {
type redirect;
dlz other;
};
```

5.12.2 Sample DLZ Driver

For guidance in the implementation of DLZ modules, the directory `contrib/dlz/example` contains a basic dynamically linkable DLZ module - i.e., one which can be loaded at runtime by the "dlopen" DLZ driver. The example sets up a single zone, whose name is passed to the module as an argument in the `dlz` statement:

```
dlz other {
database "dlopen driver.so example.nil";
};
```

In the above example, the module is configured to create a zone "example.nil", which can answer queries and AXFR requests and accept DDNS updates. At runtime, prior to any updates, the zone contains an SOA, NS, and a single A record at the apex:

```
example.nil.  3600    IN      SOA     example.nil. hostmaster.example.nil. (
                          123 900 600 86400 3600
                      )
example.nil.  3600    IN      NS      example.nil.
example.nil.  1800    IN      A       10.53.0.1
```

The sample driver can retrieve information about the querying client and alter its response on the basis of this information. To demonstrate this feature, the example driver responds to queries for "sourceaddr.‘‘zonename‘‘>/TXT" with the source address of the query. Note, however, that this record will *not* be included in AXFR or ANY responses. Normally, this feature would be used to alter responses in some other fashion, e.g., by providing different address records for a particular name depending on the network from which the query arrived.

Documentation of the DLZ module API can be found in `contrib/dlz/example/README`. This directory also contains the header file `dlz_minimal.h`, which defines the API and should be included by any dynamically linkable DLZ module.

5.13 Dynamic Database (DynDB)

Dynamic Database, or DynDB, is an extension to BIND 9 which, like DLZ (see *Dynamically Loadable Zones (DLZ)*), allows zone data to be retrieved from an external database. Unlike DLZ, a DynDB module provides a full-featured BIND zone database interface. Where DLZ translates DNS queries into real-time database lookups, resulting in relatively poor query performance, and is unable to handle DNSSEC-signed data due to its limited API, a DynDB module can pre-load an in-memory database from the external data source, providing the same performance and functionality as zones served natively by BIND.

A DynDB module supporting LDAP has been created by Red Hat and is available from https://pagure.io/bind-dyndb-ldap.

A sample DynDB module for testing and developer guidance is included with the BIND source code, in the directory `bin/tests/system/dyndb/driver`.

5.13.1 Configuring DynDB

A DynDB database is configured with a `dyndb` statement in `named.conf`:

```
dyndb example "driver.so" {
    parameters
};
```

The file `driver.so` is a DynDB module which implements the full DNS database API. Multiple `dyndb` statements can be specified, to load different drivers or multiple instances of the same driver. Zones provided by a DynDB module are added to the view's zone table, and are treated as normal authoritative zones when BIND responds to queries. Zone configuration is handled internally by the DynDB module.

The parameters are passed as an opaque string to the DynDB module's initialization routine. Configuration syntax differs depending on the driver.

5.13.2 Sample DynDB Module

For guidance in the implementation of DynDB modules, the directory `bin/tests/system/dyndb/driver` contains a basic DynDB module. The example sets up two zones, whose names are passed to the module as arguments in the `dyndb` statement:

```
dyndb sample "sample.so" { example.nil. arpa. };
```

In the above example, the module is configured to create a zone "example.nil" which can answer queries and AXFR requests and accept DDNS updates. At runtime, prior to any updates, the zone contains an SOA, NS, and a single A record at the apex:

```
example.nil.  86400    IN      SOA     example.nil. example.nil. (
                                              0 28800 7200 604800 86400
                                      )
example.nil.  86400    IN      NS      example.nil.
example.nil.  86400    IN      A       127.0.0.1
```

When the zone is updated dynamically, the DynDB module determines whether the updated RR is an address (i.e., type A or AAAA) and if so, it automatically updates the corresponding PTR record in a reverse zone. Note that updates are not stored permanently; all updates are lost when the server is restarted.

5.14 Catalog Zones

A "catalog zone" is a special DNS zone that contains a list of other zones to be served, along with their configuration parameters. Zones listed in a catalog zone are called "member zones." When a catalog zone is loaded or transferred to a secondary server which supports this functionality, the secondary server creates the member zones automatically. When the catalog zone is updated (for example, to add or delete member zones, or change their configuration parameters), those changes are immediately put into effect. Because the catalog zone is a normal DNS zone, these configuration changes can be propagated using the standard AXFR/IXFR zone transfer mechanism.

Catalog zones' format and behavior are specified as an Internet draft for interoperability among DNS implementations. The latest revision of the DNS catalog zones draft can be found here: https://datatracker.ietf.org/doc/draft-toorop-dnsop-dns-catalog-zones/.

5.14.1 Principle of Operation

Normally, if a zone is to be served by a secondary server, the `named.conf` file on the server must list the zone, or the zone must be added using `rndc addzone`. In environments with a large number of secondary servers, and/or where the zones being served are changing frequently, the overhead involved in maintaining consistent zone configuration on all the secondary servers can be significant.

A catalog zone is a way to ease this administrative burden: it is a DNS zone that lists member zones that should be served by secondary servers. When a secondary server receives an update to the catalog zone, it adds, removes, or reconfigures member zones based on the data received.

To use a catalog zone, it must first be set up as a normal zone on both the primary and secondary servers that are configured to use it. It must also be added to a `catalog-zones` list in the `options` or `view` statement in `named.conf`. This is comparable to the way a policy zone is configured as a normal zone and also listed in a `response-policy` statement.

To use the catalog zone feature to serve a new member zone:

- Set up the the member zone to be served on the primary as normal. This can be done by editing `named.conf` or by running `rndc addzone`.
- Add an entry to the catalog zone for the new member zone. This can be done by editing the catalog zone's master file and running `rndc reload`, or by updating the zone using `nsupdate`.

The change to the catalog zone is propagated from the primary to all secondaries using the normal AXFR/IXFR mechanism. When the secondary receives the update to the catalog zone, it detects the entry for the new member zone, creates an instance of that zone on the secondary server, and points that instance to the `masters` specified in the catalog zone data. The newly created member zone is a normal secondary zone, so BIND immediately initiates a transfer of zone contents from the primary. Once complete, the secondary starts serving the member zone.

Removing a member zone from a secondary server requires only deleting the member zone's entry in the catalog zone; the change to the catalog zone is propagated to the secondary server using the normal AXFR/IXFR transfer mechanism. The secondary server, on processing the update, notices that the member zone has been removed, stops serving the zone, and removes it from its list of configured zones. However, removing the member zone from the primary server must be done by editing the configuration file or running `rndc delzone`.

5.14.2 Configuring Catalog Zones

Catalog zones are configured with a `catalog-zones` statement in the `options` or `view` section of `named.conf`. For example:

```
catalog-zones {
    zone "catalog.example"
         default-masters { 10.53.0.1; }
         in-memory no
         zone-directory "catzones"
         min-update-interval 10;
};
```

This statement specifies that the zone `catalog.example` is a catalog zone. This zone must be properly configured in the same view. In most configurations, it would be a secondary zone.

The options following the zone name are not required, and may be specified in any order.

`default-masters` This option defines the default primaries for member zones listed in a catalog zone, and can be overridden by options within a catalog zone. If no such options are included, then member zones transfer their contents from the servers listed in this option.

`in-memory` This option, if set to `yes`, causes member zones to be stored only in memory. This is functionally equivalent to configuring a secondary zone without a `file` option. The default is `no`; member zones' content is stored locally in a file whose name is automatically generated from the view name, catalog zone name, and member zone name.

`zone-directory` This option causes local copies of member zones' master files to be stored in the specified directory, if `in-memory` is not set to `yes`. The default is to store zone files in the server's working directory. A non-absolute pathname in `zone-directory` is assumed to be relative to the working directory.

`min-update-interval` This option sets the minimum interval between processing of updates to catalog zones, in seconds. If an update to a catalog zone (for example, via IXFR) happens less than `min-update-interval` seconds after the most recent update, the changes are not carried out until this interval has elapsed. The default is 5 seconds.

Catalog zones are defined on a per-view basis. Configuring a non-empty `catalog-zones` statement in a view automatically turns on `allow-new-zones` for that view. This means that `rndc addzone` and `rndc delzone` also work in any view that supports catalog zones.

5.14.3 Catalog Zone Format

A catalog zone is a regular DNS zone; therefore, it must have a single `SOA` and at least one `NS` record.

A record stating the version of the catalog zone format is also required. If the version number listed is not supported by the server, then a catalog zone may not be used by that server.

```
catalog.example.     IN SOA . . 2016022901 900 600 86400 1
catalog.example.     IN NS nsexample.
version.catalog.example.     IN TXT "1"
```

Note that this record must have the domain name version.catalog-zone-name. The data stored in a catalog zone is indicated by the domain name label immediately before the catalog zone domain.

Catalog zone options can be set either globally for the whole catalog zone or for a single member zone. Global options override the settings in the configuration file, and member zone options override global options.

Global options are set at the apex of the catalog zone, e.g.:

```
masters.catalog.example.    IN AAAA 2001:db8::1
```

BIND currently supports the following options:

- A simple `masters` definition:

  ```
  masters.catalog.example.    IN A 192.0.2.1
  ```

  This option defines a primary server for the member zones - it can be either an A or AAAA record. If multiple primaries are set, the order in which they are used is random.

- A `masters` with a TSIG key defined:

  ```
  label.masters.catalog.example.     IN A 192.0.2.2
  label.masters.catalog.example.     IN TXT "tsig_key_name"
  ```

  This option defines a primary server for the member zone with a TSIG key set. The TSIG key must be configured in the configuration file. `label` can be any valid DNS label.

- `allow-query` and `allow-transfer` ACLs:

  ```
  allow-query.catalog.example.   IN APL 1:10.0.0.1/24
  allow-transfer.catalog.example.    IN APL !1:10.0.0.1/32 1:10.0.0.0/24
  ```

  These options are the equivalents of `allow-query` and `allow-transfer` in a zone declaration in the `named.conf` configuration file. The ACL is processed in order; if there is no match to any rule, the default policy is to deny access. For the syntax of the APL RR, see **RFC 3123**.

A member zone is added by including a `PTR` resource record in the `zones` sub-domain of the catalog zone. The record label is a `SHA-1` hash of the member zone name in wire format. The target of the PTR record is the member zone name. For example, to add the member zone `domain.example`:

```
5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN PTR domain.example.
```

The hash is necessary to identify options for a specific member zone. The member zone-specific options are defined the same way as global options, but in the member zone subdomain:

```
masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN A 192.0.2.2
label.masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN AAAA␣
↪2001:db8::2
label.masters.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN TXT
↪"tsig_key"
allow-query.5960775ba382e7a4e09263fc06e7c00569b6a05c.zones.catalog.example. IN APL 1:10.
↪0.0.0/24
```

Options defined for a specific zone override the global options defined in the catalog zone. These in turn override the global options defined in the `catalog-zones` statement in the configuration file.

Note that none of the global records for an option are inherited if any records are defined for that option for the specific zone. For example, if the zone had a `masters` record of type A but not AAAA, it would *not* inherit the type AAAA record from the global option.

5.15 IPv6 Support in BIND 9

BIND 9 fully supports all currently defined forms of IPv6 name-to-address and address-to-name lookups. It also uses IPv6 addresses to make queries when running on an IPv6-capable system.

For forward lookups, BIND 9 supports only AAAA records. **RFC 3363** deprecated the use of A6 records, and client-side support for A6 records was accordingly removed from BIND 9. However, authoritative BIND 9 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, BIND 9 supports the traditional "nibble" format used in the `ip6.arpa` domain, as well as the older, deprecated `ip6.int` domain. Older versions of BIND 9 supported the "binary label" (also known as "bitstring") format, but support of binary labels has been completely removed per **RFC 3363**. Many applications in BIND 9 do not understand the binary label format at all anymore, and return an error if one is given. In particular, an authoritative BIND 9 name server will not load a zone file containing binary labels.

For an overview of the format and structure of IPv6 addresses, see *IPv6 Addresses (AAAA)*.

5.15.1 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.

5.15.2 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. )
```

CHAPTER 6

BIND 9 Security Considerations

6.1 Access Control Lists

Access Control Lists (ACLs) are address match lists that can be set up and nicknamed for future use in `allow-notify`, `allow-query`, `allow-query-on`, `allow-recursion`, `blackhole`, `allow-transfer`, `match-clients`, etc.

ACLs give users finer control over who can access the name server, without cluttering up config files with huge lists of IP addresses.

It is a *good idea* to use ACLs, and to control access. Limiting access to the server by outside parties can help prevent spoofing and denial of service (DoS) attacks against the server.

ACLs match clients on the basis of up to three characteristics: 1) The client's IP address; 2) the TSIG or SIG(0) key that was used to sign the request, if any; and 3) an address prefix encoded in an EDNS Client Subnet option, if any.

Here is an example of ACLs based on client addresses:

```
// Set up an ACL named "bogusnets" that blocks
// 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 {
  ...
  ...
```

(continues on next page)

(continued from previous page)

```
  allow-query { our-nets; };
  allow-recursion { our-nets; };
  ...
  blackhole { bogusnets; };
  ...
};

zone "example.com" {
  type master;
  file "m/example.com";
  allow-query { any; };
};
```

This allows authoritative queries for `example.com` from any address, but recursive queries only from the networks specified in `our-nets`, and no queries at all from the networks specified in `bogusnets`.

In addition to network addresses and prefixes, which are matched against the source address of the DNS request, ACLs may include `key` elements, which specify the name of a TSIG or SIG(0) key.

When BIND 9 is built with GeoIP support, ACLs can also be used for geographic access restrictions. This is done by specifying an ACL element of the form: `geoip db database field value`.

The `field` parameter indicates which field to search for a match. Available fields are `country`, `region`, `city`, `continent`, `postal` (postal code), `metro` (metro code), `area` (area code), `tz` (timezone), `isp`, `asnum`, and `domain`.

`value` is the value to search for within the database. A string may be quoted if it contains spaces or other special characters. An `asnum` search for autonomous system number can be specified using the string "ASNNNN" or the integer NNNN. If a `country` search is specified with a string that is two characters long, it must be a standard ISO-3166-1 two-letter country code; otherwise it is interpreted as the full name of the country. Similarly, if `region` is the search term and the string is two characters long, it is treated as a standard two-letter state or province abbreviation; otherwise, it is treated as the full name of the state or province.

The `database` field indicates which GeoIP database to search for a match. In most cases this is unnecessary, because most search fields can only be found in a single database. However, searches for `continent` or `country` can be answered from either the `city` or `country` databases, so for these search types, specifying a `database` forces the query to be answered from that database and no other. If a `database` is not specified, these queries are first answered from the `city` database if it is installed, and then from the `country` database if it is installed. Valid database names are `country`, `city`, `asnum`, `isp`, and `domain`.

Some example GeoIP ACLs:

```
geoip country US;
geoip country JP;
geoip db country country Canada;
geoip region WA;
geoip city "San Francisco";
geoip region Oklahoma;
geoip postal 95062;
geoip tz "America/Los_Angeles";
geoip org "Internet Systems Consortium";
```

ACLs use a "first-match" logic rather than "best-match": if an address prefix matches an ACL element, then that ACL is considered to have matched even if a later element would have matched more specifically. For

example, the ACL `{ 10/8; !10.0.0.1; }` would actually match a query from 10.0.0.1, because the first element indicates that the query should be accepted, and the second element is ignored.

When using "nested" ACLs (that is, ACLs included or referenced within other ACLs), a negative match of a nested ACL tells the containing ACL to continue looking for matches. This enables complex ACLs to be constructed, in which multiple client characteristics can be checked at the same time. For example, to construct an ACL which allows queries only when it originates from a particular network *and* only when it is signed with a particular key, use:

```
allow-query { !{ !10/8; any; }; key example; };
```

Within the nested ACL, any address that is *not* in the 10/8 network prefix is rejected, which terminates processing of the ACL. Any address that *is* in the 10/8 network prefix is accepted, but this causes a negative match of the nested ACL, so the containing ACL continues processing. The query is accepted if it is signed by the key `example`, and rejected otherwise. The ACL, then, only matches when *both* conditions are true.

6.2 `Chroot` and `Setuid`

On Unix servers, it is possible to run BIND in a *chrooted* environment (using the `chroot()` function) by specifying the `-t` option for `named`. This can help improve system security by placing BIND in a "sandbox," which limits the damage done if a server is compromised.

Another useful feature in the Unix version of BIND is the ability to run the daemon as an unprivileged user (`-u` user). We suggest running as an unprivileged user when using the `chroot` feature.

Here is an example command line to load BIND in a `chroot` sandbox, `/var/named`, and to run `named setuid` to user 202:

```
/usr/local/sbin/named -u 202 -t /var/named
```

6.2.1 The `chroot` Environment

For a `chroot` environment to work properly in a particular directory (for example, `/var/named`), the environment must include everything BIND needs to run. From BIND's point of view, `/var/named` is the root of the filesystem; the values of options like `directory` and `pid-file` must be adjusted to account for this.

Unlike with earlier versions of BIND, `named` does *not* typically need to be compiled statically, nor do shared libraries need to be installed under the new root. However, depending on the operating system, it may be necessary to set up locations such as `/dev/zero`, `/dev/random`, `/dev/log`, and `/etc/localtime`.

6.2.2 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 where BIND should write.

Note: If the `named` daemon is running as an unprivileged user, it cannot bind to new restricted ports if the server is reloaded.

6.3 Dynamic Update Security

Access to the dynamic update facility should be strictly limited. In earlier versions of BIND, the only way to do this was 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. This method is insecure since the source address of the update UDP packet is easily forged. Also note that if the IP addresses allowed by the `allow-update` option include the address of a secondary server which performs forwarding of dynamic updates, the primary can be trivially attacked by sending the update to the secondary, which forwards it to the primary with its own source IP address - causing the primary to approve it without question.

For these reasons, we strongly recommend that updates be cryptographically authenticated 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 new `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 zone. This way, the top-level zone containing critical data such as the IP addresses of public web and mail servers need not allow dynamic updates at all.

CHAPTER 7

Troubleshooting

7.1 Common Problems

7.1.1 It's Not Working; How Can I Figure Out What's Wrong?

The best solution to installation and configuration issues is to take preventive measures by setting up logging files beforehand. The log files provide hints and information that can be used to identify anything that went wrong and fix the problem.

7.1.2 EDNS Compliance Issues

EDNS (Extended DNS) is a standard that was first specified in 1999. It is required for DNSSEC validation, DNS COOKIE options, and other features. There are broken and outdated DNS servers and firewalls still in use which misbehave when queried with EDNS; for example, they may drop EDNS queries rather than replying with FORMERR. BIND and other recursive name servers have traditionally employed workarounds in this situation, retrying queries in different ways and eventually falling back to plain DNS queries without EDNS.

Such workarounds cause unnecessary resolution delays, increase code complexity, and prevent deployment of new DNS features. In February 2019, all major DNS software vendors removed these workarounds; see https://dnsflagday.net/2019 for further details. This change was implemented in BIND as of release 9.14.0.

As a result, some domains may be non-resolvable without manual intervention. In these cases, resolution can be restored by adding `server` clauses for the offending servers, specifying `edns no` or `send-cookie no`, depending on the specific noncompliance.

To determine which `server` clause to use, run the following commands to send queries to the authoritative servers for the broken domain:

```
dig soa <zone> @<server> +dnssec
dig soa <zone> @<server> +dnssec +nocookie
dig soa <zone> @<server> +noedns
```

If the first command fails but the second succeeds, the server most likely needs `send-cookie no`. If the first two fail but the third succeeds, then the server needs EDNS to be fully disabled with `edns no`.

Please contact the administrators of noncompliant domains and encourage them to upgrade their broken DNS servers.

7.2 Incrementing and Changing the Serial Number

Zone serial numbers are just numbers — they are not date related. However, many people set them to a number that represents a date, usually of the form YYYYMMDDRR. Occasionally they will make a mistake and set the serial number to a date in the future, then try to correct it by setting it 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 secondary server is lower than the serial number on the primary, the secondary server attempts to update its copy of the zone.

Setting the serial number to a lower number on the primary server than the one on the secondary server means that the secondary 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 secondaries have updated to the new zone serial number, then reset it to the desired number and reload the zone again.

7.3 Where Can I Get Help?

The BIND-users mailing list, at https://lists.isc.org/mailman/listinfo/bind-users, is an excellent resource for peer user support. In addition, ISC maintains a Knowledgebase of helpful articles at https://kb.isc.org.

Internet Systems Consortium (ISC) offers annual support agreements for BIND 9, ISC DHCP and Kea DHCP. All paid support contracts include advance security notifications; some levels include service level agreements (SLAs), premium software features, and increased priority on bug fixes and feature requests.

Please contact info@isc.org or visit https://www.isc.org/contact/ for more information.

CHAPTER 8

Release Notes

Contents

8.1 Introduction

BIND 9.16 is a stable branch of BIND. This document summarizes significant changes since the last production release on that branch. Please see the file CHANGES for a more detailed list of changes and bug fixes.

8.2 Note on Version Numbering

As of BIND 9.13/9.14, BIND has adopted the "odd-unstable/even-stable" release numbering convention. BIND 9.16 contains new features that were added during the BIND 9.15 development process. Henceforth, the 9.16 branch will be limited to bug fixes, and new feature development will proceed in the unstable 9.17 branch.

8.3 Supported Platforms

To build on UNIX-like systems, BIND requires support for POSIX.1c threads (IEEE Std 1003.1c-1995), the Advanced Sockets API for IPv6 (**RFC 3542**), and standard atomic operations provided by the C compiler.

The libuv asynchronous I/O library and the OpenSSL cryptography library must be available for the target platform. A PKCS#11 provider can be used instead of OpenSSL for Public Key cryptography (i.e., DNSSEC signing and validation), but OpenSSL is still required for general cryptography operations such as hashing and random number generation.

More information can be found in the `PLATFORMS.md` file that is included in the source distribution of BIND 9. If your compiler and system libraries provide the above features, BIND 9 should compile and run. If that is not the case, the BIND development team will generally accept patches that add support for systems that are still supported by their respective vendors.

8.4 Download

The latest versions of BIND 9 software can always be found at https://www.isc.org/download/. There you will find additional information about each release, source code, and pre-compiled versions for Microsoft Windows operating systems.

8.5 Notes for BIND 9.16.6

8.5.1 Security Fixes

- It was possible to trigger an assertion failure by sending a specially crafted large TCP DNS message. This was disclosed in CVE-2020-8620.

  ISC would like to thank Emanuel Almeida of Cisco Systems, Inc. for bringing this vulnerability to our attention. [GL #1996]

- `named` could crash after failing an assertion check in certain query resolution scenarios where QNAME minimization and forwarding were both enabled. To prevent such crashes, QNAME minimization is now always disabled for a given query resolution process, if forwarders are used at any point. This was disclosed in CVE-2020-8621.

  ISC would like to thank Joseph Gullo for bringing this vulnerability to our attention. [GL #1997]

- It was possible to trigger an assertion failure when verifying the response to a TSIG-signed request. This was disclosed in CVE-2020-8622.

  ISC would like to thank Dave Feldman, Jeff Warren, and Joel Cunningham of Oracle for bringing this vulnerability to our attention. [GL #2028]

- When BIND 9 was compiled with native PKCS#11 support, it was possible to trigger an assertion failure in code determining the number of bits in the PKCS#11 RSA public key with a specially crafted packet. This was disclosed in CVE-2020-8623.

  ISC would like to thank Lyu Chiy for bringing this vulnerability to our attention. [GL #2037]

- `update-policy` rules of type `subdomain` were incorrectly treated as `zonesub` rules, which allowed keys used in `subdomain` rules to update names outside of the specified subdomains. The problem was fixed by making sure `subdomain` rules are again processed as described in the ARM. This was disclosed in CVE-2020-8624.

  ISC would like to thank Joop Boonen of credativ GmbH for bringing this vulnerability to our attention. [GL #2055]

8.5.2 New Features

- A new configuration option `stale-cache-enable` has been introduced to enable or disable keeping stale answers in cache. [GL #1712]

8.5.3 Feature Changes

- BIND's cache database implementation has been updated to use a faster hash function with better distribution. In addition, the effective `max-cache-size` (configured explicitly, defaulting to a value based on system memory or set to `unlimited`) now pre-allocates fixed-size hash tables. This prevents interruption to query resolution when the hash table sizes need to be increased. [GL #1775]
- Resource records received with 0 TTL are no longer kept in the cache to be used for stale answers. [GL #1829]

8.5.4 Bug Fixes

- Wildcard RPZ passthru rules could incorrectly be overridden by other rules that were loaded from RPZ zones which appeared later in the `response-policy` statement. This has been fixed. [GL #1619]
- The IPv6 Duplicate Address Detection (DAD) mechanism could inadvertently prevent `named` from binding to new IPv6 interfaces, by causing multiple route socket messages to be sent for each IPv6 address. `named` monitors for new interfaces to `bind()` to when it is configured to listen on `any` or on a specific range of addresses. New IPv6 interfaces can be in a "tentative" state before they are fully available for use. When DAD is in use, two messages are emitted by the route socket: one when the interface first appears and then a second one when it is fully "up." An attempt by `named` to `bind()` to the new interface prematurely would fail, causing it thereafter to ignore that address/interface. The problem was worked around by setting the `IP_FREEBIND` option on the socket and trying to `bind()` to each IPv6 address again if the first `bind()` call for that address failed with `EADDRNOTAVAIL`. [GL #2038]
- Addressed an error in recursive clients stats reporting which could cause underflow, and even negative statistics. There were occasions when an incoming query could trigger a prefetch for some eligible RRset, and if the prefetch code were executed before recursion, no increment in recursive clients stats would take place. Conversely, when processing the answers, if the recursion code were executed before the prefetch, the same counter would be decremented without a matching increment. [GL #1719]
- The introduction of KASP support inadvertently caused the second field of `sig-validity-interval` to always be calculated in hours, even in cases when it should have been calculated in days. This has been fixed. (Thanks to Tony Finch.) [GL !3735]
- LMDB locking code was revised to make `rndc reconfig` work properly on FreeBSD and with LMDB >= 0.9.26. [GL #1976]

8.6 Notes for BIND 9.16.5

8.6.1 New Features

- New `rndc` command `rndc dnssec -status` shows the current DNSSEC policy and keys in use, the key states, and rollover status. [GL #1612]

8.6.2 Bug Fixes

- A race condition could occur if a TCP socket connection was closed while `named` was waiting for a recursive response. The attempt to send a response over the closing connection triggered an assertion failure in the function `isc__nm_tcpdns_send()`. [GL #1937]
- A race condition could occur when `named` attempted to use a UDP interface that was shutting down. This triggered an assertion failure in `uv__udp_finish_close()`. [GL #1938]
- Fix assertion failure when server was under load and root zone had not yet been loaded. [GL #1862]
- `named` could crash when cleaning dead nodes in `lib/dns/rbtdb.c` that were being reused. [GL #1968]
- `named` crashed on shutdown when a new `rndc` connection was received during shutdown. This has been fixed. [GL #1747]
- The DS RRset returned by `dns_keynode_dsset()` was used in a non-thread-safe manner. This could result in an INSIST being triggered. [GL #1926]
- Properly handle missing `kyua` command so that `make check` does not fail unexpectedly when CMocka is installed, but Kyua is not. [GL #1950]
- The `primary` and `secondary` keywords, when used as parameters for `check-names`, were not processed correctly and were being ignored. [GL #1949]
- `rndc dnstap -roll <value>` did not limit the number of saved files to `<value>`. [GL !3728]
- The validator could fail to accept a properly signed RRset if an unsupported algorithm appeared earlier in the DNSKEY RRset than a supported algorithm. It could also stop if it detected a malformed public key. [GL #1689]
- The `blackhole` ACL was inadvertently disabled for client queries. Blocked IP addresses were not used for upstream queries but queries from those addresses could still be answered. [GL #1936]

8.7 Notes for BIND 9.16.4

8.7.1 Security Fixes

- It was possible to trigger an assertion when attempting to fill an oversized TCP buffer. This was disclosed in CVE-2020-8618. [GL #1850]
- It was possible to trigger an INSIST failure when a zone with an interior wildcard label was queried in a certain pattern. This was disclosed in CVE-2020-8619. [GL #1111] [GL #1718]

8.7.2 New Features

- Documentation was converted from DocBook to reStructuredText. The BIND 9 ARM is now generated using Sphinx and published on Read the Docs. Release notes are no longer available as a separate document accompanying a release. [GL #83]
- `named` and `named-checkzone` now reject master zones that have a DS RRset at the zone apex. Attempts to add DS records at the zone apex via UPDATE will be logged but otherwise ignored. DS records belong in the parent zone, not at the zone apex. [GL #1798]
- `dig` and other tools can now print the Extended DNS Error (EDE) option when it appears in a request or a response. [GL #1835]

8.7.3 Feature Changes

- The default value of `max-stale-ttl` has changed from 1 week to 12 hours. This option controls how long `named` retains expired RRsets in cache as a potential mitigation mechanism, should there be a problem with one or more domains. Note that cache content retention is independent of whether stale answers are used in response to client queries (`stale-answer-enable yes|no` and `rndc serve-stale on|off`). Serving of stale answers when the authoritative servers are not responding must be explicitly enabled, whereas the retention of expired cache content takes place automatically on all versions of BIND 9 that have this feature available. [GL #1877]

  > **Warning:** This change may be significant for administrators who expect that stale cache content will be automatically retained for up to 1 week. Add option `max-stale-ttl 1w;` to `named.conf` to keep the previous behavior of `named`.

- `listen-on-v6 { any; }` creates a separate socket for each interface. Previously, just one socket was created on systems conforming to **RFC 3493** and **RFC 3542**. This change was introduced in BIND 9.16.0, but it was accidentally omitted from documentation. [GL #1782]

8.7.4 Bug Fixes

- When fully updating the NSEC3 chain for a large zone via IXFR, a temporary loss of performance could be experienced on the secondary server when answering queries for nonexistent data that required DNSSEC proof of non-existence (in other words, queries that required the server to find and to return NSEC3 data). The unnecessary processing step that was causing this delay has now been removed. [GL #1834]
- `named` could crash with an assertion failure if the name of a database node was looked up while the database was being modified. [GL #1857]
- A possible deadlock in `lib/isc/unix/socket.c` was fixed. [GL #1859]
- Previously, `named` did not destroy some mutexes and conditional variables in netmgr code, which caused a memory leak on FreeBSD. This has been fixed. [GL #1893]
- A data race in `lib/dns/resolver.c:log_formerr()` that could lead to an assertion failure was fixed. [GL #1808]
- Previously, `provide-ixfr no;` failed to return up-to-date responses when the serial number was greater than or equal to the current serial number. [GL #1714]
- A bug in dnssec-policy keymgr was fixed, where the check for the existence of a given key's successor would incorrectly return `true` if any other key in the keyring had a successor. [GL #1845]
- With dnssec-policy, when creating a successor key, the "goal" state of the current active key (the predecessor) was not changed and thus never removed from the zone. [GL #1846]
- `named-checkconf -p` could include spurious text in `server-addresses` statements due to an uninitialized DSCP value. This has been fixed. [GL #1812]
- The ARM has been updated to indicate that the TSIG session key is generated when named starts, regardless of whether it is needed. [GL #1842]

8.8 Notes for BIND 9.16.3

8.8.1 Known Issues

- BIND crashes on startup when linked against libuv 1.36. This issue is related to `recvmmsg()` support in libuv, which was first included in libuv 1.35. The problem was addressed in libuv 1.37, but the relevant libuv code change requires a special flag to be set during library initialization in order for `recvmmsg()` support to be enabled. This BIND release sets that special flag when required, so `recvmmsg()` support is now enabled when BIND is compiled against either libuv 1.35 or libuv 1.37+; libuv 1.36 is still not usable with BIND. [GL #1761] [GL #1797]

8.8.2 Feature Changes

- BIND 9 no longer sets receive/send buffer sizes for UDP sockets, relying on system defaults instead. [GL #1713]
- The default rwlock implementation has been changed back to the native BIND 9 rwlock implementation. [GL #1753]
- The native PKCS#11 EdDSA implementation has been updated to PKCS#11 v3.0 and thus made operational again. Contributed by Aaron Thompson. [GL !3326]
- The OpenSSL ECDSA implementation has been updated to support PKCS#11 via OpenSSL engine (see engine_pkcs11 from libp11 project). [GL #1534]
- The OpenSSL EdDSA implementation has been updated to support PKCS#11 via OpenSSL engine. Please note that an EdDSA-capable OpenSSL engine is required and thus this code is only a proof-of-concept for the time being. Contributed by Aaron Thompson. [GL #1763]
- Message IDs in inbound AXFR transfers are now checked for consistency. Log messages are emitted for streams with inconsistent message IDs. [GL #1674]
- The zone timers are now exported to the statistics channel. For the primary zones, only the loaded time is exported. For the secondary zones, the exported timers also include expire and refresh times. Contributed by Paul Frieden, Verizon Media. [GL #1232]

8.8.3 Bug Fixes

- A bug in dnstap initialization could prevent some dnstap data from being logged, especially on recursive resolvers. [GL #1795]
- When running on a system with support for Linux capabilities, `named` drops root privileges very soon after system startup. This was causing a spurious log message, `unable to set effective uid to 0: Operation not permitted`, which has now been silenced. [GL #1042] [GL #1090]
- When `named-checkconf` was run, it would sometimes incorrectly set its exit code. It reflected only the status of the last view found; any errors found for other configured views were not reported. Thanks to Graham Clinch. [GL #1807]
- When built without LMDB support, `named` failed to restart after a zone with a double quote (") in its name was added with `rndc addzone`. Thanks to Alberto Fernández. [GL #1695]

8.9 Notes for BIND 9.16.2

8.9.1 Security Fixes

- DNS rebinding protection was ineffective when BIND 9 is configured as a forwarding DNS server. Found and responsibly reported by Tobias Klein. [GL #1574]

8.9.2 Known Issues

- We have received reports that in some circumstances, receipt of an IXFR can cause the processing of queries to slow significantly. Some of these were related to RPZ processing, which has been fixed in this release (see below). Others appear to occur where there are NSEC3-related changes (such as an operator changing the NSEC3 salt used in the hash calculation). These are being investigated. [GL #1685]

8.9.3 Feature Changes

- The previous DNSSEC sign statistics used lots of memory. The number of keys to track is reduced to four per zone, which should be enough for 99% of all signed zones. [GL #1179]

8.9.4 Bug Fixes

- When an RPZ policy zone was updated via zone transfer and a large number of records was deleted, `named` could become nonresponsive for a short period while deleted names were removed from the RPZ summary database. This database cleanup is now done incrementally over a longer period of time, reducing such delays. [GL #1447]
- When trying to migrate an already-signed zone from `auto-dnssec maintain` to one based on `dnssec-policy`, the existing keys were immediately deleted and replaced with new ones. As the key rollover timing constraints were not being followed, it was possible that some clients would not have been able to validate responses until all old DNSSEC information had timed out from caches. BIND now looks at the time metadata of the existing keys and incorporates it into its DNSSEC policy operation. [GL #1706]

8.10 Notes for BIND 9.16.1

8.10.1 Known Issues

- UDP network ports used for listening can no longer simultaneously be used for sending traffic. An example configuration which triggers this issue would be one which uses the same address:port pair for `listen-on(-v6)` statements as for `notify-source(-v6)` or `transfer-source(-v6)`. While this issue affects all operating systems, it only triggers log messages (e.g. "unable to create dispatch for reserved port") on some of them. There are currently no plans to make such a combination of settings work again.

8.10.2 Feature Changes

- The system-provided POSIX Threads read-write lock implementation is now used by default instead of the native BIND 9 implementation. Please be aware that glibc versions 2.26 through 2.29 had a bug that could cause BIND 9 to deadlock. A fix was released in glibc 2.30, and most current Linux distributions have patched or updated glibc, with the notable exception of Ubuntu 18.04 (Bionic) which is a work in progress. If you are running on an affected operating system, compile BIND 9 with `--disable-pthread-rwlock` until a fixed version of glibc is available. [GL !3125]

8.10.3 Bug Fixes

- Fixed re-signing issues with inline zones which resulted in records being re-signed late or not at all.

8.11 Notes for BIND 9.16.0

Note: this section only lists changes from BIND 9.14 (the previous stable branch of BIND).

8.11.1 New Features

- A new asynchronous network communications system based on `libuv` is now used by `named` for listening for incoming requests and responding to them. This change will make it easier to improve performance and implement new protocol layers (for example, DNS over TLS) in the future. [GL #29]
- The new `dnssec-policy` option allows the configuration of a key and signing policy (KASP) for zones. This option enables `named` to generate new keys as needed and automatically roll both ZSK and KSK keys. (Note that the syntax for this statement differs from the DNSSEC policy used by `dnssec-keymgr`.) [GL #1134]
- In order to clarify the configuration of DNSSEC keys, the `trusted-keys` and `managed-keys` statements have been deprecated, and the new `trust-anchors` statement should now be used for both types of key.

  When used with the keyword `initial-key`, `trust-anchors` has the same behavior as `managed-keys`, i.e., it configures a trust anchor that is to be maintained via RFC 5011.

  When used with the new keyword `static-key`, `trust-anchors` has the same behavior as `trusted-keys`, i.e., it configures a permanent trust anchor that will not automatically be updated. (This usage is not recommended for the root key.) [GL #6]
- Two new keywords have been added to the `trust-anchors` statement: `initial-ds` and `static-ds`. These allow the use of trust anchors in DS format instead of DNSKEY format. DS format allows trust anchors to be configured for keys that have not yet been published; this is the format used by IANA when announcing future root keys.

  As with the `initial-key` and `static-key` keywords, `initial-ds` configures a dynamic trust anchor to be maintained via RFC 5011, and `static-ds` configures a permanent trust anchor. [GL #6] [GL #622]
- `dig`, `mdig` and `delv` can all now take a `+yaml` option to print output in a detailed YAML format. [GL #1145]
- `dig` now has a new command line option: `+[no]unexpected`. By default, `dig` won't accept a reply from a source other than the one to which it sent the query. Add the `+unexpected` argument to enable it to process replies from unexpected sources. [RT #44978]

- `dig` now accepts a new command line option, `+[no]expandaaaa`, which causes the IPv6 addresses in AAAA records to be printed in full 128-bit notation rather than the default RFC 5952 format. [GL #765]
- Statistics channel groups can now be toggled. [GL #1030]

8.11.2 Feature Changes

- When static and managed DNSSEC keys were both configured for the same name, or when a static key was used to configure a trust anchor for the root zone and `dnssec-validation` was set to the default value of `auto`, automatic RFC 5011 key rollovers would be disabled. This combination of settings was never intended to work, but there was no check for it in the parser. This has been corrected, and it is now a fatal configuration error. [GL #868]
- DS and CDS records are now generated with SHA-256 digests only, instead of both SHA-1 and SHA-256. This affects the default output of `dnssec-dsfromkey`, the `dsset` files generated by `dnssec-signzone`, the DS records added to a zone by `dnssec-signzone` based on `keyset` files, the CDS records added to a zone by `named` and `dnssec-signzone` based on "sync" timing parameters in key files, and the checks performed by `dnssec-checkds`. [GL #1015]
- `named` will now log a warning if a static key is configured for the root zone. [GL #6]
- A SipHash 2-4 based DNS Cookie (RFC 7873) algorithm has been added and made default. Old non-default HMAC-SHA based DNS Cookie algorithms have been removed, and only the default AES algorithm is being kept for legacy reasons. This change has no operational impact in most common scenarios. [GL #605]

  If you are running multiple DNS servers (different versions of BIND 9 or DNS servers from multiple vendors) responding from the same IP address (anycast or load-balancing scenarios), make sure that all the servers are configured with the same DNS Cookie algorithm and same Server Secret for the best performance.
- The information from the `dnssec-signzone` and `dnssec-verify` commands is now printed to standard output. The standard error output is only used to print warnings and errors, and in case the user requests the signed zone to be printed to standard output with the `-f -` option. A new configuration option `-q` has been added to silence all output on standard output except for the name of the signed zone. [GL #1151]
- The DNSSEC validation code has been refactored for clarity and to reduce code duplication. [GL #622]
- Compile-time settings enabled by the `--with-tuning=large` option for `configure` are now in effect by default. Previously used default compile-time settings can be enabled by passing `--with-tuning=small` to `configure`. [GL !2989]
- JSON-C is now the only supported library for enabling JSON support for BIND statistics. The `configure` option has been renamed from `--with-libjson` to `--with-json-c`. Set the `PKG_CONFIG_PATH` environment variable accordingly to specify a custom path to the `json-c` library, as the new `configure` option does not take the library installation path as an optional argument. [GL #855]
- `./configure` no longer sets `--sysconfdir` to `/etc` or `--localstatedir` to `/var` when `--prefix` is not specified and the aforementioned options are not specified explicitly. Instead, Autoconf's defaults of `$prefix/etc` and `$prefix/var` are respected. [GL #658]

8.11.3 Removed Features

- The `dnssec-enable` option has been obsoleted and no longer has any effect. DNSSEC responses are always enabled if signatures and other DNSSEC data are present. [GL #866]
- DNSSEC Lookaside Validation (DLV) is now obsolete. The `dnssec-lookaside` option has been marked as deprecated; when used in `named.conf`, it will generate a warning but will otherwise be ignored. All code enabling the use of lookaside validation has been removed from the validator, `delv`, and the DNSSEC tools. [GL #7]
- The `cleaning-interval` option has been removed. [GL !1731]

8.12 License

BIND 9 is open source software licensed under the terms of the Mozilla Public License, version 2.0 (see the `LICENSE` file for the full text).

The license requires that if you make changes to BIND and distribute them outside your organization, those changes must be published under the same license. It does not require that you publish or disclose anything other than the changes you have made to our software. This requirement does not affect anyone who is using BIND, with or without modifications, without redistributing it, nor anyone redistributing BIND without changes.

Those wishing to discuss license compliance may contact ISC at https://www.isc.org/contact/.

8.13 End of Life

The end of life date for BIND 9.16 has not yet been determined. At some point in the future, BIND 9.16 will be designated as an Extended Support Version (ESV). Until then, the current ESV is BIND 9.11, which will be supported until at least December 2021. See https://kb.isc.org/docs/aa-00896 for details of ISC's software support policy.

8.14 Thank You

Thank you to everyone who assisted us in making this release possible.

CHAPTER 9

A Brief History of the DNS and BIND

Although the Domain Name System "officially" began in 1984 with the publication of **RFC 920**, the core of the new system was described in 1983 in **RFC 882** and **RFC 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 Cherenson, 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 and its predecessor, the Internet Software Consortium, with support 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.

BIND versions 4 and 8 are officially deprecated. No additional development is done on BIND version 4 or BIND version 8.

BIND development work is made possible today by the sponsorship of corporations who purchase professional support services from ISC (https://www.isc.org/contact/) and/or donate to our mission, and by the tireless efforts of numerous individuals.

CHAPTER 10

General DNS Reference Information

10.1 IPv6 Addresses (AAAA)

IPv6 addresses are 128-bit identifiers, for interfaces and sets of interfaces, which were introduced in the DNS to facilitate scalable Internet routing. There are three types of addresses: *Unicast*, an identifier for a single interface; *Anycast*, an identifier for a set of interfaces; and *Multicast*, an identifier for a set of interfaces. Here we describe the global Unicast address scheme. For more information, see **RFC 3587**, "IPv6 Global Unicast Address Format."

IPv6 unicast addresses consist of a *global routing prefix*, a *subnet identifier*, and an *interface identifier*.

The global routing prefix is provided by the upstream provider or ISP, and roughly corresponds to the IPv4 *network* section of the address range. The subnet identifier is for local subnetting, much like subnetting an IPv4 /16 network into /24 subnets. The interface identifier is the address of an individual interface on a given network; in IPv6, addresses belong to interfaces rather than to machines.

The subnetting capability of IPv6 is much more flexible than that of IPv4: subnetting can be carried out on bit boundaries, in much the same way as Classless InterDomain Routing (CIDR), and the DNS PTR representation ("nibble" format) makes setting up reverse zones easier.

The interface identifier must be unique on the local link, and is usually generated automatically by the IPv6 implementation, although it is usually possible to override the default setting if necessary. A typical IPv6 address might look like: `2001:db8:201:9:a00:20ff:fe81:2b32`

IPv6 address specifications often contain long strings of zeros, so the architects have included a shorthand for specifying them. The double colon (`::`) indicates the longest possible string of zeros that can fit, and can be used only once in an address.

10.2 Bibliography (and Suggested Reading)

10.2.1 Request for Comments (RFCs)

BIND 9 strives for strict compliance with IETF standards. To the best of our knowledge, BIND 9 complies with the following RFCs, with the caveats and exceptions listed in the numbered notes below. Many of these RFCs were written by current or former ISC staff members. The list is non-exhaustive.

Specification documents for the Internet protocol suite, including the DNS, are published as part of the Request for Comments (RFCs) series of technical notes. The standards themselves are defined by the Internet Engineering Task Force (IETF) and the Internet Engineering Steering Group (IESG). RFCs can be viewed online at: https://datatracker.ietf.org/doc/.

Some of these RFCs, though DNS-related, are not concerned with implementing software.

10.3 Internet Standards

RFC 1034 - P. Mockapetris. *Domain Names — Concepts and Facilities.* November 1987.

RFC 1035 - P. Mockapetris. *Domain Names — Implementation and Specification.* November 1987. [1] [2]

RFC 1123 - R. Braden. *Requirements for Internet Hosts - Application and Support.* October 1989.

RFC 3596 - S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. *DNS Extensions to Support IP Version 6.* October 2003.

RFC 5011 - M. StJohns. *Automated Updates of DNS Security (DNSSEC) Trust Anchors.*

RFC 6891 - J. Damas, M. Graff, and P. Vixie. *Extension Mechanisms for DNS (EDNS(0)).* April 2013.

10.4 Proposed Standards

RFC 1982 - R. Elz and R. Bush. *Serial Number Arithmetic.* August 1996.

RFC 1995 - M. Ohta. *Incremental Zone Transfer in DNS.* August 1996.

RFC 1996 - P. Vixie. *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY).* August 1996.

RFC 2136 - P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE).* April 1997.

RFC 2163 - A. Allocchio. *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM).* January 1998.

RFC 2181 - R. Elz and R. Bush. *Clarifications to the DNS Specification.* July 1997.

RFC 2308 - M. Andrews. *Negative Caching of DNS Queries (DNS NCACHE).* March 1998.

RFC 2539 - D. Eastlake, 3rd. *Storage of Diffie-Hellman Keys in the Domain Name System (DNS).* March 1999.

RFC 2782 - A. Gulbrandsen, P. Vixie, and L. Esibov. *A DNS RR for Specifying the Location of Services (DNS SRV).* February 2000.

RFC 2845 - P. Vixie, O. Gudmundsson, D. Eastlake, 3rd, and B. Wellington. *Secret Key Transaction Authentication for DNS (TSIG).* May 2000.

RFC 2930 - D. Eastlake, 3rd. *Secret Key Establishment for DNS (TKEY RR).* September 2000.

RFC 2931 - D. Eastlake, 3rd. *DNS Request and Transaction Signatures (SIG(0)s).* September 2000. [3]

RFC 3007 - B. Wellington. *Secure Domain Name System (DNS) Dynamic Update.* November 2000.

RFC 3110 - D. Eastlake, 3rd. *RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS).* May 2001.

RFC 3225 - D. Conrad. *Indicating Resolver Support of DNSSEC.* December 2001.

RFC 3226 - O. Gudmundsson. *DNSSEC and IPv6 A6 Aware Server/Resolver Message Size Requirements.* December 2001.

RFC 3492 - A. Costello. *Punycode: A Bootstring Encoding of Unicode for Internationalized Domain Names in Applications (IDNA).* March 2003.

RFC 3597 - A. Gustafsson. *Handling of Unknown DNS Resource Record (RR) Types.* September 2003.

RFC 3645 - S. Kwan, P. Garg, J. Gilroy, L. Esibov, J. Westhead, and R. Hall. *Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG).* October 2003.

RFC 4025 - M. Richardson. *A Method for Storing IPsec Keying Material in DNS.* March 2005.

RFC 4033 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *DNS Security Introduction and Requirements.* March 2005. [4]

RFC 4034 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *Resource Records for the DNS Security Extensions.* March 2005.

RFC 4035 - R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *Protocol Modifications for the DNS Security Extensions.* March 2005.

RFC 4255 - J. Schlyter and W. Griffin. *Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints.* January 2006.

RFC 4343 - D. Eastlake, 3rd. *Domain Name System (DNS) Case Insensitivity Clarification.* January 2006.

RFC 4398 - S. Josefsson. *Storing Certificates in the Domain Name System (DNS).* March 2006.

RFC 4470 - S. Weiler and J. Ihren. *Minimally Covering NSEC Records and DNSSEC On-line Signing.* April 2006. [5]

RFC 4509 - W. Hardaker. *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs).* May 2006.

RFC 4592 - E. Lewis. *The Role of Wildcards in the Domain Name System.* July 2006.

RFC 4635 - D. Eastlake, 3rd. *HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers.* August 2006.

RFC 4701 - M. Stapp, T. Lemon, and A. Gustafsson. *A DNS Resource Record (RR) for Encoding Dynamic Host Configuration Protocol (DHCP) Information (DHCID RR).* October 2006.

RFC 4955 - D. Blacka. *DNS Security (DNSSEC) Experiments.* July 2007. [6]

RFC 5001 - R. Austein. *DNS Name Server Identifier (NSID) Option.* August 2007.

RFC 5155 - B. Laurie, G. Sisson, R. Arends, and D. Blacka. *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence.* March 2008.

RFC 5452 - A. Hubert and R. van Mook. *Measures for Making DNS More Resilient Against Forged Answers.* January 2009. [7]

RFC 5702 - J. Jansen. *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC.* October 2009.

RFC 5936 - E. Lewis and A. Hoenes, Ed. *DNS Zone Transfer Protocol (AXFR).* June 2010.

RFC 5952 - S. Kawamura and M. Kawashima. *A Recommendation for IPv6 Address Text Representation.* August 2010.

RFC 6052 - C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li. *IPv6 Addressing of IPv4/IPv6 Translators.* October 2010.

RFC 6147 - M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers.* April 2011. [8]

RFC 6594 - O. Sury. *Use of the SHA-256 Algorithm with RSA, Digital Signature Algorithm (DSA), and Elliptic Curve DSA (ECDSA) in SSHFP Resource Records.* April 2012.

RFC 6604 - D. Eastlake, 3rd. *xNAME RCODE and Status Bits Clarification.* April 2012.

RFC 6605 - P. Hoffman and W. C. A. Wijngaards. *Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC.* April 2012. [9]

RFC 6672 - S. Rose and W. Wijngaards. *DNAME Redirection in the DNS.* June 2012.

RFC 6698 - P. Hoffman and J. Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA.* August 2012.

RFC 6725 - S. Rose. *DNS Security (DNSSEC) DNSKEY Algorithm IANA Registry Updates.* August 2012. [10]

RFC 6840 - S. Weiler, Ed., and D. Blacka, Ed. *Clarifications and Implementation Notes for DNS Security (DNSSEC).* February 2013. [11]

RFC 7216 - M. Thomson and R. Bellis. *Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS.* April 2014.

RFC 7344 - W. Kumari, O. Gudmundsson, and G. Barwood. *Automating DNSSEC Delegation Trust Maintenance.* September 2014. [12]

RFC 7477 - W. Hardaker. *Child-to-Parent Synchronization in DNS.* March 2015.

RFC 7766 - J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels. *DNS Transport over TCP - Implementation Requirements.* March 2016.

RFC 7828 - P. Wouters, J. Abley, S. Dickinson, and R. Bellis. *The edns-tcp-keepalive EDNS0 Option.* April 2016.

RFC 7830 - A. Mayrhofer. *The EDNS(0) Padding Option.* May 2016. [13]

RFC 8080 - O. Sury and R. Edmonds. *Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC.* February 2017.

RFC 8482 - J. Abley, O. Gudmundsson, M. Majkowski, and E. Hunt. *Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY.* January 2019.

RFC 8490 - R. Bellis, S. Cheshire, J. Dickinson, S. Dickinson, T. Lemon, and T. Pusateri. *DNS Stateful Operations.* March 2019.

RFC 8624 - P. Wouters and O. Sury. *Algorithm Implementation Requirements and Usage Guidance for DNSSEC.* June 2019.

RFC 8749 - W. Mekking and D. Mahoney. *Moving DNSSEC Lookaside Validation (DLV) to Historic Status.* March 2020.

10.5 Informational RFCs

RFC 1535 - E. Gavron. *A Security Problem and Proposed Correction With Widely Deployed DNS Software.* October 1993.

RFC 1536 - A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller. *Common DNS Implementation Errors and Suggested Fixes.* October 1993.

RFC 1591 - J. Postel. *Domain Name System Structure and Delegation.* March 1994.

RFC 1706 - B. Manning and R. Colella. *DNS NSAP Resource Records.* October 1994.

RFC 1713 - A. Romao. *Tools for DNS Debugging.* November 1994.

RFC 1794 - T. Brisco. *DNS Support for Load Balancing.* April 1995.

RFC 1912 - D. Barr. *Common DNS Operational and Configuration Errors.* February 1996.

RFC 2230 - R. Atkinson. *Key Exchange Delegation Record for the DNS.* November 1997.

RFC 2352 - O. Vaughan. *A Convention for Using Legal Names as Domain Names.* May 1998.

RFC 2825 - IAB and L. Daigle. *A Tangled Web: Issues of I18N, Domain Names, and the Other Internet Protocols.* May 2000.

RFC 2826 - Internet Architecture Board. *IAB Technical Comment on the Unique DNS Root.* May 2000.

RFC 3071 - J. Klensin. *Reflections on the DNS, RFC 1591, and Categories of Domains.* February 2001.

RFC 3258 - T. Hardie. *Distributing Authoritative Name Servers via Shared Unicast Addresses.* April 2002.

RFC 3363 - R. Bush, A. Durand, B. Fink, O. Gudmundsson, and T. Hain. *Representing Internet Protocol Version 6 (IPv6) Addresses in the Domain Name System (DNS).* August 2002. [14]

RFC 3493 - R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens. *Basic Socket Interface Extensions for IPv6.* March 2003.

RFC 3496 - A. G. Malis and T. Hsiao. *Protocol Extension for Support of Asynchronous Transfer Mode (ATM) Service Class-aware Multiprotocol Label Switching (MPLS) Traffic Engineering.* March 2003.

RFC 3833 - D. Atkins and R. Austein. *Threat Analysis of the Domain Name System (DNS).* August 2004.

RFC 4074 - Y. Morishita and T. Jinmei. *Common Misbehavior Against DNS Queries for IPv6 Addresses.* June 2005.

RFC 4892 - S. Woolf and D. Conrad. *Requirements for a Mechanism Identifying a Name Server Instance.* June 2007.

RFC 6781 - O. Kolkman, W. Mekking, and R. Gieben. *DNSSEC Operational Practices, Version 2.* December 2012.

RFC 7043 - J. Abley. *Resource Records for EUI-48 and EUI-64 Addresses in the DNS.* October 2013.

RFC 7129 - R. Gieben and W. Mekking. *Authenticated Denial of Existence in the DNS.* February 2014.

RFC 7553 - P. Faltstrom and O. Kolkman. *The Uniform Resource Identifier (URI) DNS Resource Record.* June 2015.

RFC 7583 - S. Morris, J. Ihren, J. Dickinson, and W. Mekking. *DNSSEC Key Rollover Timing Considerations.* October 2015.

10.6 Experimental RFCs

RFC 1183 - C. F. Everhart, L. A. Mamakos, R. Ullmann, P. Mockapetris. *New DNS RR Definitions.* October 1990.

RFC 1464 - R. Rosenbaum. *Using the Domain Name System to Store Arbitrary String Attributes.* May 1993.

RFC 1712 - C. Farrell, M. Schulze, S. Pleitner, and D. Baldoni. *DNS Encoding of Geographical Location.* November 1994.

RFC 1876 - C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. *A Means for Expressing Location Information in the Domain Name System.* January 1996.

RFC 2345 - J. Klensin, T. Wolf, and G. Oglesby. *Domain Names and Company Name Retrieval.* May 1998.

RFC 2540 - D. Eastlake, 3rd. *Detached Domain Name System (DNS) Information.* March 1999.

RFC 3123 - P. Koch. *A DNS RR Type for Lists of Address Prefixes (APL RR).* June 2001.

RFC 6742 - RJ Atkinson, SN Bhatti, U. St. Andrews, and S. Rose. *DNS Resource Records for the Identifier-Locator Network Protocol (ILNP).* November 2012.

RFC 7314 - M. Andrews. *Extension Mechanisms for DNS (EDNS) EXPIRE Option.* July 2014.

RFC 7929 - P. Wouters. *DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP.* August 2016.

10.7 Best Current Practice RFCs

RFC 2219 - M. Hamilton and R. Wright. *Use of DNS Aliases for Network Services.* October 1997.

RFC 2317 - H. Eidnes, G. de Groot, and P. Vixie. *Classless IN-ADDR.ARPA Delegation.* March 1998.

RFC 2606 - D. Eastlake, 3rd and A. Panitz. *Reserved Top Level DNS Names.* June 1999. [15]

RFC 3901 - A. Durand and J. Ihren. *DNS IPv6 Transport Operational Guidelines.* September 2004.

RFC 5625 - R. Bellis. *DNS Proxy Implementation Guidelines.* August 2009.

RFC 6303 - M. Andrews. *Locally Served DNS Zones.* July 2011.

RFC 7793 - M. Andrews. *Adding 100.64.0.0/10 Prefixes to the IPv4 Locally-Served DNS Zones Registry.* May 2016.

10.8 Historic RFCs

RFC 2874 - M. Crawford and C. Huitema. *DNS Extensions to Support IPv6 Address Aggregation and Renumbering.* July 2000. [4]

RFC 4431 - M. Andrews and S. Weiler. *The DNSSEC Lookaside Validation (DLV) DNS Resource Record.* February 2006.

10.9 RFCs of Type "Unknown"

RFC 1033 - M. Lottor. *Domain Administrators Operations Guide.* November 1987.

RFC 1101 - P. Mockapetris. *DNS Encoding of Network Names and Other Types.* April 1989.

10.10 Obsoleted and Unimplemented Experimental RFCs

RFC 974 - C. Partridge. *Mail Routing and the Domain System.* January 1986.

RFC 1521 - N. Borenstein and N. Freed. *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies.* September 1993 [16]

RFC 1537 - P. Beertema. *Common DNS Data File Configuration Errors.* October 1993.

RFC 1750 - D. Eastlake, 3rd, S. Crocker, and J. Schiller. *Randomness Recommendations for Security.* December 1994.

RFC 2010 - B. Manning and P. Vixie. *Operational Criteria for Root Name Servers.* October 1996.

RFC 2052 - A. Gulbrandsen and P. Vixie. *A DNS RR for Specifying the Location of Services.* October 1996.

RFC 2065 - D. Eastlake, 3rd and C. Kaufman. *Domain Name System Security Extensions.* January 1997.

RFC 2137 - D. Eastlake, 3rd. *Secure Domain Name System Dynamic Update.* April 1997.

RFC 2168 - R. Daniel and M. Mealling. *Resolution of Uniform Resource Identifiers Using the Domain Name System.* June 1997.

RFC 2240 - O. Vaughan. *A Legal Basis for Domain Name Allocation.* November 1997.

RFC 2535 - D. Eastlake, 3rd. *Domain Name System Security Extensions.* March 1999. [17] [18]

RFC 2537 - D. Eastlake, 3rd. *RSA/MD5 KEYs and SIGs in the Domain Name System (DNS).* March 1999.

RFC 2538 - D. Eastlake, 3rd and O. Gudmundsson. *Storing Certificates in the Domain Name System (DNS).* March 1999.

RFC 2671 - P. Vixie. *Extension Mechanisms for DNS (EDNS0).* August 1999.

RFC 2672 - M. Crawford. *Non-Terminal DNS Name Redirection.* August 1999.

RFC 2673 - M. Crawford. *Binary Labels in the Domain Name System.* August 1999.

RFC 2915 - M. Mealling and R. Daniel. *The Naming Authority Pointer (NAPTR) DNS Resource Record.* September 2000.

RFC 2929 - D. Eastlake, 3rd, E. Brunner-Williams, and B. Manning. *Domain Name System (DNS) IANA Considerations.* September 2000.

RFC 3008 - B. Wellington. *Domain Name System Security (DNSSEC) Signing Authority.* November 2000.

RFC 3090 - E. Lewis. *DNS Security Extension Clarification on Zone Status.* March 2001.

RFC 3152 - R. Bush. *Delegation of IP6.ARPA.* August 2001.

RFC 3445 - D. Massey and S. Rose. *Limiting the Scope of the KEY Resource Record (RR).* December 2002.

RFC 3490 - P. Faltstrom, P. Hoffman, and A. Costello. *Internationalizing Domain Names in Applications (IDNA).* March 2003. [19]

RFC 3491 - P. Hoffman and M. Blanchet. *Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN).* March 2003. [19]

RFC 3655 - B. Wellington and O. Gudmundsson. *Redefinition of DNS Authenticated Data (AD) Bit.* November 2003.

RFC 3658 - O. Gudmundsson. *Delegation Signer (DS) Resource Record (RR).* December 2003.

RFC 3755 - S. Weiler. *Legacy Resolver Compatibility for Delegation Signer (DS).* May 2004.

RFC 3757 - O. Kolkman, J. Schlyter, and E. Lewis. *Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag.* May 2004.

RFC 3845 - J. Schlyter. *DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format.* August 2004.

RFC 4294 - J. Loughney, Ed. *IPv6 Node Requirements.* [20]

RFC 4408 - M. Wong and W. Schlitt. *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1.* April 2006.

RFC 5966 - R. Bellis. *DNS Transport Over TCP - Implementation Requirements.* August 2010.

RFC 6844 - P. Hallam-Baker and R. Stradling. *DNS Certification Authority Authorization (CAA) Resource Record.* January 2013.

RFC 6944 - S. Rose. *Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status.* April 2013.

10.11 RFCs No Longer Supported in BIND 9

RFC 2536 - D. Eastlake, 3rd. *DSA KEYs and SIGs in the Domain Name System (DNS).* March 1999.

10.11.1 Notes

[1] Queries to zones that have failed to load return SERVFAIL rather than a non-authoritative response. This is considered a feature.

[2] CLASS ANY queries are not supported. This is considered a feature.

[3] When receiving a query signed with a SIG(0), the server is only able to verify the signature if it has the key in its local authoritative data; it cannot do recursion or validation to retrieve unknown keys.

[4] Compliance is with loading and serving of A6 records only. A6 records were moved to the experimental category by **RFC 3363**.

[5] Minimally covering NSEC records are accepted but not generated.

[6] BIND 9 interoperates with correctly designed experiments.

[7] `named` only uses ports to extend the ID space; addresses are not used.

[8] Section 5.5 does not match reality. `named` uses the presence of DO=1 to detect if validation may be occurring. CD has no bearing on whether validation occurs.

[9] Compliance is conditional on the OpenSSL library being linked against a supporting ECDSA.

[10] RSAMD5 support has been removed. See **RFC 6944**.

[11] Section 5.9 - Always set CD=1 on queries. This is *not* done, as it prevents DNSSEC from working correctly through another recursive server.

When talking to a recursive server, the best algorithm is to send CD=0 and then send CD=1 iff SERVFAIL is returned, in case the recursive server has a bad clock and/or bad trust anchor. Alternatively, one can send CD=1 then CD=0 on validation failure, in case the recursive server is under attack or there is stale/bogus authoritative data.

[12] Updating of parent zones is not yet implemented.

[13] `named` does not currently encrypt DNS requests, so the PAD option is accepted but not returned in responses.

[14] Section 4 is ignored.

[15] This does not apply to DNS server implementations.

[16] Only the Base 64 encoding specification is supported.

[17] Wildcard records are not supported in DNSSEC secure zones.

[18] Servers authoritative for secure zones being resolved by BIND 9 must support EDNS0 (RFC2671), and must return all relevant SIGs and NXTs in responses, rather than relying on the resolving server to perform separate queries for missing SIGs and NXTs.

[19] BIND 9 requires `--with-idn` to enable entry of IDN labels within dig, host, and nslookup at compile time. ACE labels are supported everywhere with or without `--with-idn`.

[20] Section 5.1 - DNAME records are fully supported.

10.11.2 Internet Drafts

Internet Drafts (IDs) are rough-draft working documents of the Internet Engineering Task Force (IETF). They are, in essence, RFCs in the preliminary stages of development. Implementors are cautioned not to regard IDs as archival, and they should not be quoted or cited in any formal documents unless accompanied by the disclaimer that they are "works in progress." IDs have a lifespan of six months, after which they are deleted unless updated by their authors.

10.11.3 Other Documents About BIND

Paul Albitz and Cricket Liu. *DNS and BIND.* Copyright 1998 Sebastopol, CA: O'Reilly and Associates.

CHAPTER 11

Manual Pages

11.1 rndc.conf - rndc configuration file

11.1.1 Synopsis

`rndc.conf`

11.1.2 Description

`rndc.conf` is the configuration file for `rndc`, the BIND 9 name server control utility. This file has a similar structure and syntax to `named.conf`. Statements are enclosed in braces and terminated with a semi-colon. Clauses in the statements are also semi-colon terminated. The usual comment styles are supported:

C style: /* */

C++ style: // to end of line

Unix style: # to end of line

`rndc.conf` is much simpler than `named.conf`. The file uses three statements: an options statement, a server statement and a key statement.

The `options` statement contains five clauses. The `default-server` clause is followed by the name or address of a name server. This host will be used when no name server 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 server's commands and responses. The `default-port` clause is followed by the port to connect to on the remote name server. 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 will be used to connect. The `default-source-address` and `default-source-address-v6` clauses which can be used to set the IPv4 and IPv6 source addresses respectively.

After the `server` keyword, the server statement includes a string which is the hostname or address for a name server. 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.

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; currently only HMAC-MD5 (for compatibility), HMAC-SHA1, HMAC-SHA224, HMAC-SHA256 (default), HMAC-SHA384 and HMAC-SHA512 are supported. This is followed by a secret clause which contains the base-64 encoding of the algorithm's authentication key. The base-64 string is enclosed in double quotes.

There are two common ways to generate the base-64 string for the secret. The BIND 9 program `rndc-confgen` can be used to generate a random key, or the `mmencode` program, also known as `mimencode`, can be used to generate a base-64 string from known input. `mmencode` does not ship with BIND 9 but is available on many systems. See the EXAMPLE section for sample command lines for each.

11.1.3 Example

```
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          "6FMfj43Osz4lyb24OIe2iGEz9lf1llJO+lz";
};
```

```
key testkey {
  algorithm   hmac-sha256;
  secret      "R3HI8P6BKw9ZwXwN3VZKuQ==";
};
```

In the above example, `rndc` will by default use the server at localhost (127.0.0.1) 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 base-64 encoding of the HMAC-SHA256 secret enclosed in double quotes.

If `rndc -s testserver` is used then `rndc` will connect to server on localhost port 5353 using the key testkey.

To generate a random secret with `rndc-confgen`:

`rndc-confgen`

A complete `rndc.conf` file, including the randomly generated key, will be written to the standard output. Commented-out `key` and `controls` statements for `named.conf` are also printed.

To generate a base-64 secret with `mmencode`:

```
echo "known plaintext for a secret" | mmencode
```

11.1.4 Name Server Configuration

The name server must be configured to accept rndc connections and to recognize the key specified in the `rndc.conf` file, using the controls statement in `named.conf`. See the sections on the `controls` statement in the BIND 9 Administrator Reference Manual for details.

11.1.5 See Also

rndc(8), *rndc-confgen(8)*, *mmencode(1)*, BIND 9 Administrator Reference Manual.

11.2 rndc - name server control utility

11.2.1 Synopsis

rndc [**-b** source-address] [**-c** config-file] [**-k** key-file] [**-s** server] [**-p** port] [**-q**] [**-r**] [**-V**] [**-y** key_id] [[**-4**] | [**-6**]] {command}

11.2.2 Description

`rndc` controls the operation of a name server. It supersedes the `ndc` utility that was provided in old BIND releases. If `rndc` is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

`rndc` communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. In the current versions of `rndc` and `named`, the only supported authentication algorithms are HMAC-MD5 (for compatibility), HMAC-SHA1, HMAC-SHA224, HMAC-SHA256 (default), HMAC-SHA384 and HMAC-SHA512. They use a shared secret on each end of the connection. This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server.

`rndc` reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

11.2.3 Options

-4 Use IPv4 only.

-6 Use IPv6 only.

-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 configuration file instead of the default, `/etc/rndc.conf`.

-k key-file Use key-file as the key file instead of the default, `/etc/rndc.key`. The key in `/etc/rndc.key` will be used to authenticate commands sent to the server if the config-file does not exist.

-s server server is the name or address of the server which matches a server statement in the configuration file for `rndc`. If no server is supplied on the command line, the host named by the default-server clause in the options statement of the `rndc` configuration file will be used.

-p port Send commands to TCP port port instead of BIND 9's default control channel port, 953.

-q Quiet mode: Message text returned by the server will not be printed except when there is an error.

-r Instructs `rndc` to print the result code returned by `named` after executing the requested command (e.g., ISC_R_SUCCESS, ISC_R_FAILURE, etc).

-V Enable verbose logging.

-y key_id Use the key key_id from the configuration file. key_id must be known by `named` with the same algorithm and secret string in order for control message validation to succeed. If no key_id is specified, `rndc` will first look for a key clause in the server statement of the server being used, or if no server statement is present for that host, then the default-key clause of the options statement. Note that the configuration file contains shared secrets which are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

11.2.4 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` option 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 called `viewname.nzf` (or, if *named(8)* is compiled with liblmdb, an LMDB database file called `viewname.nzd`). viewname is the name of the view, unless the view name contains characters that are incompatible with use as a file name, in which case a cryptographic hash of the view name is used instead. 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.

This sample `addzone` command would add the zone `example.com` to the default view:

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

(Note the brackets and semi-colon around the zone configuration text.)

See also `rndc delzone` and `rndc modzone`.

`delzone` [**-clean**] ***zone*** [***class*** [***view***]] Delete a zone while the server is running.

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 `rndc delzone` command.)

If the zone was originally added via `rndc addzone`, then it will be removed permanently. However, if it was originally configured in `named.conf`, then that original configuration is still in place; when the server is restarted or reconfigured, the zone will come back. To remove it permanently, it must also be removed from `named.conf`

See also `rndc addzone` and `rndc modzone`.

`dnssec` [**-status** ***zone*** [***class*** [***view***]] Show the DNSSEC signing state for the specified zone. Requires the zone to have a "dnssec-policy".

`dnstap ( -reopen | -roll [number] )` Close and re-open DNSTAP output files. `rndc dnstap -reopen` allows the output file to be renamed externally, so that *named(8)* can truncate and re-open it. `rndc dnstap -roll` causes the output file to be rolled automatically, similar to log files; the most recent output file has ".0" appended to its name; the previous most recent output file is moved to ".1", and so on. If number is specified, then the number of backup log files is limited to that number.

`dumpdb [-all | -cache | -zones | -adb | -bad | -fail] [view ...]` Dump the server's caches (default) and/or zones to the dump file for the specified views. If no view is specified, all views are dumped. (See the `dump-file` option in the BIND 9 Administrator Reference Manual.)

`flush` Flushes the server's cache.

`flushname name [view]` Flushes the given name from the view's DNS cache and, if applicable, from the view's nameserver address database, bad server cache and SERVFAIL cache.

`flushtree name [view]` Flushes the given name, and all of its subdomains, from the view's DNS cache, address database, bad server cache, and SERVFAIL 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.

See also `rndc thaw`.

`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 *named(8)*'s process id is returned. This allows an external process to determine when *named(8)* had completed halting.

See also `rndc stop`.

`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 `rndc sign`, 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 zone is configured with a `dnssec-policy`, or the `auto-dnssec` zone option be set to `maintain`, and also requires the zone to be configured to allow dynamic DNS. (See "Dynamic Update Policies" in the Administrator Reference Manual for more details.)

`managed-keys (status | refresh | sync | destroy) [class [view]]` Inspect and control the "managed-keys" database which handles **RFC 5011** DNSSEC trust anchor maintenance. If a view is specified, these commands are applied to that view; otherwise they are applied to all views.

- When run with the `status` keyword, prints the current status of the managed-keys database.
- When run with the `refresh` keyword, forces an immediate refresh query to be sent for all the managed keys, updating the managed-keys database if any new keys are found, without waiting the normal refresh interval.
- When run with the `sync` keyword, forces an immediate dump of the managed-keys database to disk (in the file `managed-keys.bind` or (`viewname.mkeys`). This synchronizes the database with its journal file, so that the database's current contents can be inspected visually.
- When run with the `destroy` keyword, the managed-keys database is shut down and deleted, and all key maintenance is terminated. This command should be used only with extreme caution.

  Existing keys that are already trusted are not deleted from memory; DNSSEC validation can continue after this command is used. However, key maintenance operations will cease until *named(8)* is restarted or reconfigured, and all existing key maintenance state will be deleted.

Running `rndc reconfig` or restarting *named(8)* immediately after this command will cause key maintenance to be reinitialized from scratch, just as if the server were being started for the first time. This is primarily intended for testing, but it may also be used, for example, to jumpstart the acquisition of new keys in the event of a trust anchor rollover, or as a brute-force repair for key maintenance problems.

`modzone` ***zone*** [***class*** [***view***]] ***configuration*** Modify the configuration of a zone while the server is running. This command requires the `allow-new-zones` option to be set to `yes`. As with `addzone`, the configuration string specified on the command line is the zone configuration text that would ordinarily be placed in `named.conf`.

If the zone was originally added via `rndc addzone`, the configuration changes will be recorded permanently and will still be in effect after the server is restarted or reconfigured. However, if it was originally configured in `named.conf`, then that original configuration is still in place; when the server is restarted or reconfigured, the zone will revert to its original configuration. To make the changes permanent, it must also be modified in `named.conf`

See also `rndc addzone` and `rndc delzone`.

`notify` ***zone*** [***class*** [***view***]] Resend NOTIFY messages for the zone.

`notrace` Sets the server's debugging level to 0.

See also `rndc trace`.

`nta` [**(-class** ***class*** **| -dump | -force | -remove | -lifetime** ***duration***)] ***domain*** [***view***] Sets a DNSSEC negative trust anchor (NTA) for `domain`, with a lifetime of `duration`. The default lifetime is configured in `named.conf` via the `nta-lifetime` option, and defaults to one hour. The lifetime cannot exceed one week.

A negative trust anchor selectively disables DNSSEC validation for zones that are known to be failing because of misconfiguration rather than an attack. When data to be validated is at or below an active NTA (and above any other configured trust anchors), *named(8)* will abort the DNSSEC validation process and treat the data as insecure rather than bogus. This continues until the NTA's lifetime is elapsed.

NTAs persist across restarts of the *named(8)* server. The NTAs for a view are saved in a file called `name.nta`, where name is the name of the view, or if it contains characters that are incompatible with use as a file name, a cryptographic hash generated from the name of the view.

An existing NTA can be removed by using the `-remove` option.

An NTA's lifetime can be specified with the `-lifetime` option. TTL-style suffixes can be used to specify the lifetime in seconds, minutes, or hours. If the specified NTA already exists, its lifetime will be updated to the new value. Setting `lifetime` to zero is equivalent to `-remove`.

If the `-dump` is used, any other arguments are ignored, and a list of existing NTAs is printed (note that this may include NTAs that are expired but have not yet been cleaned up).

Normally, *named(8)* will periodically test to see whether data below an NTA can now be validated (see the `nta-recheck` option in the Administrator Reference Manual for details). If data can be validated, then the NTA is regarded as no longer necessary, and will be allowed to expire early. The `-force` overrides this behavior and forces an NTA to persist for its entire lifetime, regardless of whether data could be validated if the NTA were not present.

The view class can be specified with `-class`. The default is class `IN`, which is the only class for which DNSSEC is currently supported.

All of these options can be shortened, i.e., to `-l`, `-r`, `-d`, `-f`, and `-c`.

Unrecognized options are treated as errors. To reference a domain or view name that begins with a hyphen, use a double-hyphen on the command line to indicate the end of options.

`querylog` [(***on*** | ***off***)] Enable or disable query logging. (For backward compatibility, this command can also be used without an argument to toggle query logging on and off.)

Query logging can also be enabled by explicitly directing the `queries category` to a `channel` in the `logging` section of `named.conf` or by specifying `querylog yes;` in the `options` section of `named.conf`.

`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` when there is a large number of zones because it avoids the need to examine the modification times of the zones files.

`recursing` 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` option.)

`refresh` ***zone*** [***class*** [***view***]] Schedule zone maintenance for the given zone.

`reload` Reload configuration file and zones.

`reload` ***zone*** [***class*** [***view***]] Reload the given zone.

`retransfer` ***zone*** [***class*** [***view***]] Retransfer the given slave zone from the master server.

If the zone is configured to use `inline-signing`, 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 performing a full `reconfig` or waiting for the `interface-interval` timer.

`secroots` [**-**] [***view ...***] Dump the security roots (i.e., trust anchors configured via `trust-anchors`, or the `managed-keys` or `trusted-keys` statements (both deprecated), or `dnssec-validation auto`) and negative trust anchors for the specified views. If no view is specified, all views are dumped. Security roots will indicate whether they are configured as trusted keys, managed keys, or initializing managed keys (managed keys that have not yet been updated by a successful key refresh query).

If the first argument is "-", then the output is returned via the `rndc` response channel and printed to the standard output. Otherwise, it is written to the secroots dump file, which defaults to `named.secroots`, but can be overridden via the `secroots-file` option in `named.conf`.

See also `rndc managed-keys`.

`serve-stale` **(on | off | reset | status)** [***class*** [***view***]] Enable, disable, reset, or report the current status of the serving of stale answers as configured in `named.conf`.

If serving of stale answers is disabled by `rndc-serve-stale off`, then it will remain disabled even if *named(8)* is reloaded or reconfigured. `rndc serve-stale reset` restores the setting as configured in `named.conf`.

`rndc serve-stale status` will report whether serving of stale answers is currently enabled, disabled by the configuration, or disabled by `rndc`. It will also report the values of `stale-answer-ttl` and `max-stale-ttl`.

`showzone` ***zone*** [***class*** [***view***]] Print the configuration of a running zone.

See also `rndc zonestatus`.

`sign` ***zone*** [***class*** [***view***]] Fetch all DNSSEC keys for the given zone from the key directory (see the `key-directory` option in the BIND 9 Administrator Reference Manual). 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 zone is configure with a `dnssec-policy`, or that the `auto-dnssec` zone option be set to `allow` or `maintain`, and also requires the zone to be configured to allow dynamic DNS. (See "Dynamic Update Policies" in the Administrator Reference Manual for more details.)

See also `rndc loadkeys`.

`signing` [(**-list** | **-clear** ***keyid/algorithm*** | **-clear** ***all*** | **-nsec3param** (***parameters*** | **none**) | **-serial** ***value***) ***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`. `rndc 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.

`rndc signing -clear` can remove a single key (specified in the same format that `rndc 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.

`rndc 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. The `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. The `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", use: `rndc signing -nsec3param 1 0 10 FFFF zone`. To set the opt-out flag, 15 iterations, and no salt, use: `rndc signing -nsec3param 1 1 15 - zone`.

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

`rndc signing -serial value` sets the serial number of the zone to value. If the value would cause the serial number to go backwards it will be rejected. The primary use is to set the serial on inline signed zones.

`stats` Write server statistics to the statistics file. (See the `statistics-file` option in the BIND 9 Administrator Reference Manual.)

`status` Display status of the server. Note that the number of zones includes the internal `bind/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 *named(8)*'s process id is returned. This allows an external process to determine when *named(8)* had completed stopping.

See also `rndc halt`.

`sync` **-clean** [***zone*** [***class*** [***view***]]] Sync changes in the journal file for a dynamic zone to the master file. If the "-clean" option is specified, the journal file is also removed. If no zone is specified, then all zones are synced.

`tcp-timeouts` [***initial idle keepalive advertised***] When called without arguments, display the current values of the `tcp-initial-timeout`, `tcp-idle-timeout`, `tcp-keepalive-timeout` and `tcp-advertised-timeout` options. When called with arguments, update these values. This allows an administrator to make rapid adjustments when under a denial of service attack. See the descriptions of these options in the BIND 9 Administrator Reference Manual for details of their use.

`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` option 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.

See also `rndc freeze`.

`trace` Increment the servers debugging level by one.

`trace` ***level*** Sets the server's debugging level to an explicit value.

See also `rndc notrace`.

`tsig-delete` ***keyname*** [***view***] Delete a given TKEY-negotiated key from the server. (This does not apply to statically configured TSIG keys.)

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

`validation` **(on | off | status)** [***view ...***]" Enable, disable, or check the current status of DNSSEC validation. By default, validation is enabled.

The cache is flushed when validation is turned on or off to avoid using data that might differ between states.

`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.

See also `rndc showzone`.

`rndc` commands that specify zone names, such as `reload`, `retransfer` or `zonestatus`, can be ambiguous when applied to zones of type `redirect`. Redirect zones are always called ".", and can be confused with zones of type `hint` or with slaved copies of the root zone. To specify a redirect zone, use the special zone name `-redirect`, without a trailing period. (With a trailing period, this would specify a zone called "-redirect".)

11.2.5 Limitations

There is currently no way to provide the shared secret for a `key_id` without using the configuration file.

Several error messages could be clearer.

11.2.6 See Also

rndc.conf(5), *rndc-confgen(8)*, *named(8)*, *named.conf(5)*, *ndc(8)*, BIND 9 Administrator Reference Manual.

11.3 nsec3hash - generate NSEC3 hash

11.3.1 Synopsis

nsec3hash {salt} {algorithm} {iterations} {domain}

nsec3hash -r {algorithm} {flags} {iterations} {salt} {domain}

11.3.2 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.

If this command is invoked as `nsec3hash -r`, it takes arguments in an order matching the first four fields of an NSEC3 record, followed by the domain name: algorithm, flags, iterations, salt, domain. This makes it convenient to copy and paste a portion of an NSEC3 or NSEC3PARAM record into a command line to confirm the correctness of an NSEC3 hash.

11.3.3 Arguments

`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.

`flags` Provided for compatibility with NSEC3 record presentation format, but ignored since the flags do not affect the hash.

`iterations` The number of additional times the hash should be performed.

`domain` The domain name to be hashed.

11.3.4 See Also

BIND 9 Administrator Reference Manual, **RFC 5155**.

11.4 dnstap-read - print dnstap data in human-readable form

11.4.1 Synopsis

`dnstap-read` [**-m**] [**-p**] [**-x**] [**-y**] {file}

11.4.2 Description

`dnstap-read` reads `dnstap` data from a specified file and prints it in a human-readable format. By default, `dnstap` data is printed in a short summary format, but if the `-y` option is specified, then a longer and more detailed YAML format is used instead.

11.4.3 Options

-m Trace memory allocations; used for debugging memory leaks.

-p After printing the `dnstap` data, print the text form of the DNS message that was encapsulated in the `dnstap` frame.

-x After printing the `dnstap` data, print a hex dump of the wire form of the DNS message that was encapsulated in the `dnstap` frame.

-y Print `dnstap` data in a detailed YAML format.

11.4.4 See Also

named(8), *rndc(8)*, BIND 9 Administrator Reference Manual.

11.5 named-nzd2nzf - convert an NZD database to NZF text format

11.5.1 Synopsis

named-nzd2nzf {filename}

11.5.2 Description

`named-nzd2nzf` converts an NZD database to NZF format and prints it to standard output. This can be used to review the configuration of zones that were added to `named` via `rndc addzone`. It can also be used to restore the old file format when rolling back from a newer version of BIND to an older version.

11.5.3 Arguments

filename The name of the `.nzd` file whose contents should be printed.

11.5.4 See Also

BIND 9 Administrator Reference Manual.

11.6 named-journalprint - print zone journal in human-readable form

11.6.1 Synopsis

named-journalprint {journal}

11.6.2 Description

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

Journal files are automatically created by `named` when changes are made to dynamic zones (e.g., by `nsupdate`). 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.

11.6.3 See Also

named(8), *nsupdate(1)*, BIND 9 Administrator Reference Manual.

11.7 mdig - DNS pipelined lookup utility

11.7.1 Synopsis

mdig {@server} [**-f** filename] [**-h**] [**-v**] [[**-4**] | [**-6**]] [**-m**] [**-b** address] [**-p** port#] [**-c** class] [**-t** type] [**-i**] [**-x** addr] [plusopt...]

mdig {**-h**}

mdig [@server] {global-opt...} { {local-opt...} {query} ...}

11.7.2 Description

`mdig` is a multiple/pipelined query version of `dig`: instead of waiting for a response after sending each query, it begins by sending all queries. Responses are displayed in the order in which they are received, not in the order the corresponding queries were sent.

`mdig` options are a subset of the `dig` options, and are divided into "anywhere options" which can occur anywhere, "global options" which must occur before the query name (or they are ignored with a warning), and "local options" which apply to the next query on the command line.

The @server option is a mandatory global option. It is the name or IP address of the name server to query. (Unlike `dig`, this value is not retrieved from `/etc/resolv.conf`.) It can be an IPv4 address in dotted-decimal notation, an IPv6 address in colon-delimited notation, or a hostname. When the supplied `server` argument is a hostname, `mdig` resolves that name before querying the name server.

`mdig` 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`.

11.7.3 Anywhere Options

The `-f` option makes `mdig` operate in batch mode by reading a list of lookup requests to process from the file `filename`. The file contains a number of queries, one per line. Each entry in the file should be organized in the same way they would be presented as queries to `mdig` using the command-line interface.

The `-h` causes `mdig` to print the detailed help with the full list of options and exit.

The `-v` causes `mdig` to print the version number and exit.

11.7.4 Global Options

The `-4` option forces `mdig` to only use IPv4 query transport.

The `-6` option forces `mdig` to only use IPv6 query transport.

The `-b` option 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 port may be specified by appending "#<port>"

The `-m` option enables memory usage debugging.

The `-p` option is used when a non-standard port number is to be queried. `port#` is the port number that `mdig` will send its queries instead of the standard DNS port number 53. This option would be used to test a name server that has been configured to listen for queries on a non-standard port number.

The global query options are:

`+[no]additional` Display [do not display] the additional section of a reply. The default is to display it.

`+[no]all` Set or clear all display flags.

`+[no]answer` Display [do not display] the answer section of a reply. The default is to display it.

`+[no]authority` Display [do not display] the authority section of a reply. The default is to display it.

`+[no]besteffort` Attempt to display the contents of messages which are malformed. The default is to not display malformed answers.

`+[no]cl` Display [do not display] the CLASS when printing the record.

`+[no]comments` Toggle the display of comment lines in the output. The default is to print comments.

`+[no]continue` Continue on errors (e.g. timeouts).

`+[no]crypto` 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]".

`+dscp[=value]` Set the DSCP code point to be used when sending the query. Valid DSCP code points are in the range [0..63]. By default no code point is explicitly set.

`+[no]multiline` 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 `mdig` output.

`+[no]question` Print [do not print] the question section of a query when an answer is returned. The default is to print the question section as a comment.

`+[no]rrcomments` 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.

`+[no]short` Provide a terse answer. The default is to print the answer in a verbose form.

`+split=W` 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 multiline mode is active.

`+[no]tcp` Use [do not use] TCP when querying name servers. The default behavior is to use UDP.

`+[no]ttlid` Display [do not display] the TTL when printing the record.

`+[no]ttlunits` Display [do not display] the TTL in friendly human-readable time units of "s", "m", "h", "d", and "w", representing seconds, minutes, hours, days and weeks. Implies +ttlid.

`+[no]vc` Use [do not use] TCP when querying name servers. This alternate syntax to `+[no]tcp` is provided for backwards compatibility. The "vc" stands for "virtual circuit".

11.7.5 Local Options

The `-c` option sets the query class to `class`. It can be any valid query class which is supported in BIND 9. The default query class is "IN".

The `-t` option sets the query type to `type`. It can be any valid query type which is supported in BIND 9. The default query type is "A", unless the `-x` option is supplied to indicate a reverse lookup with the "PTR" query type.

Reverse lookups MDASH mapping addresses to names MDASH are simplified by the `-x` option. `addr` is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. `mdig` automatically performs a lookup for a query name like `11.12.13.10.in-addr.arpa` and sets the query type and class to PTR and IN respectively. By default, IPv6 addresses are looked up using nibble format under the IP6.ARPA domain.

The local query options are:

`+[no]aaflag` A synonym for `+[no]aaonly`.

`+[no]aaonly` Sets the "aa" flag in the query.

`+[no]adflag` Set [do not set] the AD (authentic data) bit in the query. This 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 a OPT-OUT range. AD=0 indicate that some part of the answer was insecure or not validated. This bit is set by default.

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

`+[no]cdflag` Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

`+[no]cookie=####` Send a COOKIE EDNS option, with optional value. Replaying a COOKIE from a previous response will allow the server to identify a previous client. The default is `+nocookie`.

`+[no]dnssec` Requests DNSSEC records be sent by setting the DNSSEC OK bit (DO) in the OPT record in the additional section of the query.

`+[no]edns[=#]` Specify the EDNS version to query with. Valid values are 0 to 255. Setting the EDNS version will cause a EDNS query to be sent. `+noedns` clears the remembered EDNS version. EDNS is set to 0 by default.

`+[no]ednsflags[=#]` 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.

`+[no]ednsopt[=code[:value]]` Specify EDNS option with code point `code` and optionally payload of `value` as a hexadecimal string. `+noednsopt` clears the EDNS options to be sent.

`+[no]expire` Send an EDNS Expire option.

`+[no]nsid` Include an EDNS name server ID request when sending a query.

`+[no]recurse` Toggle the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means `mdig` normally sends recursive queries.

`+retry=T` Sets the number of times to retry UDP queries to server to `T` instead of the default, 2. Unlike `+tries`, this does not include the initial query.

`+[no]subnet=addr[/prefix-length]` Send (don't send) an EDNS Client Subnet option with the specified IP address or network prefix.

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

`+timeout=T` Sets the timeout for a query to `T` seconds. The default timeout is 5 seconds for UDP transport and 10 for TCP. An attempt to set `T` to less than 1 will result in a query timeout of 1 second being applied.

`+tries=T` Sets the number of times to try UDP queries to server to `T` instead of the default, 3. If `T` is less than or equal to zero, the number of tries is silently rounded up to 1.

`+udptimeout=T` Sets the timeout between UDP query retries.

`+[no]unknownformat` Print all RDATA in unknown RR type presentation format (**RFC 3597**). The default is to print RDATA for known types in the type's presentation format.

`+[no]yaml` Print the responses in a detailed YAML format.

`+[no]zflag` Set [do not set] the last unassigned DNS header flag in a DNS query. This flag is off by default.

11.7.6 See Also

dig(1), **RFC 1035**.

11.8 named-rrchecker - syntax checker for individual DNS resource records

11.8.1 Synopsis

`named-rrchecker` [**-h**] [**-o** origin] [**-p**] [**-u**] [**-C**] [**-T**] [**-P**]

11.8.2 Description

`named-rrchecker` read a individual DNS resource record from standard input and checks if it is syntactically correct.

The `-h` prints out the help menu.

The `-o origin` option specifies a origin to be used when interpreting the record.

The `-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.

The `-u` prints out the resulting record in unknown record form.

The `-C`, `-T` and `-P` print out the known class, standard type and private type mnemonics respectively.

11.8.3 See Also

RFC 1034, **RFC 1035**, *named(8)*.

11.9 arpaname - translate IP addresses to the corresponding ARPA names

11.9.1 Synopsis

`arpaname` {*ipaddress* ...}

11.9.2 Description

`arpaname` translates IP addresses (IPv4 and IPv6) to the corresponding IN-ADDR.ARPA or IP6.ARPA names.

11.9.3 See Also

BIND 9 Administrator Reference Manual.

11.10 dnssec-revoke - set the REVOKED bit on a DNSSEC key

11.10.1 Synopsis

`dnssec-revoke` [**-hr**] [**-v** level] [**-V**] [**-K** directory] [**-E** engine] [**-f**] [**-R**] {keyfile}

11.10.2 Description

`dnssec-revoke` reads a DNSSEC key file, sets the REVOKED bit on the key as defined in **RFC 5011**, and creates a new pair of key files containing the now-revoked key.

11.10.3 Options

-h Emit usage message and exit.

-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.

-v level Sets the debugging level.

-V Prints version information.

-E engine Specifies the cryptographic hardware to use, when applicable.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-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.

11.10.4 See Also

dnssec-keygen(8), BIND 9 Administrator Reference Manual, **RFC 5011**.

11.11 dnssec-cds - change DS records for a child zone based on CDS/CDNSKEY

11.11.1 Synopsis

dnssec-cds [**-a** alg...] [**-c** class] [**-D**] {**-d** dsset-file} {**-f** child-file} [**-i** [extension]] [**-s** start-time] [**-T** ttl] [**-u**] [**-v** level] [**-V**] {domain}

11.11.2 Description

The **dnssec-cds** command changes DS records at a delegation point based on CDS or CDNSKEY records published in the child zone. If both CDS and CDNSKEY records are present in the child zone, the CDS is preferred. This enables a child zone to inform its parent of upcoming changes to its key-signing keys; by polling periodically with **dnssec-cds**, the parent can keep the DS records up to date and enable automatic rolling of KSKs.

Two input files are required. The **-f child-file** option specifies a file containing the child's CDS and/or CDNSKEY records, plus RRSIG and DNSKEY records so that they can be authenticated. The **-d path** option specifies the location of a file containing the current DS records. For example, this could be a **dsset-** file generated by **dnssec-signzone**, or the output of **dnssec-dsfromkey**, or the output of a previous run of **dnssec-cds**.

The **dnssec-cds** command uses special DNSSEC validation logic specified by **RFC 7344**. It requires that the CDS and/or CDNSKEY records are validly signed by a key represented in the existing DS records. This will typically be the pre-existing key-signing key (KSK).

For protection against replay attacks, the signatures on the child records must not be older than they were on a previous run of **dnssec-cds**. This time is obtained from the modification time of the **dsset-** file, or from the **-s** option.

To protect against breaking the delegation, **dnssec-cds** ensures that the DNSKEY RRset can be verified by every key algorithm in the new DS RRset, and that the same set of keys are covered by every DS digest type.

By default, replacement DS records are written to the standard output; with the **-i** option the input file is overwritten in place. The replacement DS records will be the same as the existing records when no change is required. The output can be empty if the CDS / CDNSKEY records specify that the child zone wants to go insecure.

Warning: Be careful not to delete the DS records when **dnssec-cds** fails!

Alternatively, **dnssec-cds -u** writes an **nsupdate** script to the standard output. You can use the **-u** and **-i** options together to maintain a **dsset-** file as well as emit an **nsupdate** script.

11.11.3 Options

-a algorithm Specify a digest algorithm to use when converting CDNSKEY records to DS records. This option can be repeated, so that multiple DS records are created for each CDNSKEY record. This option has no effect when using CDS records.

The algorithm must be one of SHA-1, SHA-256, or SHA-384. These values are case insensitive, and the hyphen may be omitted. If no algorithm is specified, the default is SHA-256.

-c class Specifies the DNS class of the zones.

-D Generate DS records from CDNSKEY records if both CDS and CDNSKEY records are present in the child zone. By default CDS records are preferred.

-d path Location of the parent DS records. The path can be the name of a file containing the DS records, or if it is a directory, `dnssec-cds` looks for a `dsset-` file for the domain inside the directory.

To protect against replay attacks, child records are rejected if they were signed earlier than the modification time of the `dsset-` file. This can be adjusted with the `-s` option.

-f child-file File containing the child's CDS and/or CDNSKEY records, plus its DNSKEY records and the covering RRSIG records so that they can be authenticated.

The EXAMPLES below describe how to generate this file.

-iextension Update the `dsset-` file in place, instead of writing DS records to the standard output.

There must be no space between the `-i` and the extension. If you provide no extension then the old `dsset-` is discarded. If an extension is present, a backup of the old `dsset-` file is kept with the extension appended to its filename.

To protect against replay attacks, the modification time of the `dsset-` file is set to match the signature inception time of the child records, provided that is later than the file's current modification time.

-s start-time Specify the date and time after which RRSIG records become acceptable. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20170827133700 denotes 13:37:00 UTC on August 27th, 2017. A time relative to the `dsset-` file is indicated with -N, which is N seconds before the file modification time. A time relative to the current time is indicated with now+N.

If no start-time is specified, the modification time of the `dsset-` file is used.

-T ttl Specifies a TTL to be used for new DS records. If not specified, the default is the TTL of the old DS records. If they had no explicit TTL then the new DS records also have no explicit TTL.

-u Write an `nsupdate` script to the standard output, instead of printing the new DS reords. The output will be empty if no change is needed.

Note: The TTL of new records needs to be specified, either in the original `dsset-` file, or with the `-T` option, or using the `nsupdate ttl` command.

-V Print version information.

-v level Sets the debugging level. Level 1 is intended to be usefully verbose for general users; higher levels are intended for developers.

domain The name of the delegation point / child zone apex.

11.11.4 Exit Status

The `dnssec-cds` command exits 0 on success, or non-zero if an error occurred.

In the success case, the DS records might or might not need to be changed.

11.11.5 Examples

Before running `dnssec-signzone`, you can ensure that the delegations are up-to-date by running `dnssec-cds` on every `dsset-` file.

To fetch the child records required by `dnssec-cds` you can invoke `dig` as in the script below. It's okay if the `dig` fails since `dnssec-cds` performs all the necessary checking.

```
for f in dsset-*
do
    d=${f#dsset-}
    dig +dnssec +noall +answer $d DNSKEY $d CDNSKEY $d CDS |
    dnssec-cds -i -f /dev/stdin -d $f $d
done
```

When the parent zone is automatically signed by `named`, you can use `dnssec-cds` with `nsupdate` to maintain a delegation as follows. The `dsset-` file allows the script to avoid having to fetch and validate the parent DS records, and it keeps the replay attack protection time.

```
dig +dnssec +noall +answer $d DNSKEY $d CDNSKEY $d CDS |
dnssec-cds -u -i -f /dev/stdin -d $f $d |
nsupdate -l
```

11.11.6 See Also

dig(1), *dnssec-settime(8)*, *dnssec-signzone(8)*, *nsupdate(1)*, BIND 9 Administrator Reference Manual, **RFC 7344**.

11.12 dnssec-keygen: DNSSEC key generation tool

11.12.1 Synopsis

dnssec-keygen [**-3**] [**-A** date/offset] [**-a** algorithm] [**-b** keysize] [**-C**] [**-c** class] [**-D** date/offset] [**-d** bits] [**-D** sync date/offset] [**-E** engine] [**-f** flag] [**-G**] [**-g** generator] [**-h**] [**-I** date/offset] [**-i** interval] [**-K** directory] [**-k** policy] [**-L** ttl] [**-l** file] [**-n** nametype] [**-P** date/offset] [**-P** sync date/offset] [**-p** protocol] [**-q**] [**-R** date/offset] [**-S** key] [**-s** strength] [**-T** rrtype] [**-t** type] [**-V**] [**-v** level] {name}

11.12.2 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**.

The `name` of the key is specified on the command line. For DNSSEC keys, this must match the name of the zone for which the key is being generated.

The `dnssec-keymgr` command acts as a wrapper around `dnssec-keygen`, generating and updating keys as needed to enforce defined security policies such as key rollover scheduling. Using `dnssec-keymgr` may be preferable to direct use of `dnssec-keygen`.

11.12.3 Options

-3 Use an NSEC3-capable algorithm to generate a DNSSEC key. If this option is used with an algorithm that has both NSEC and NSEC3 versions, then the NSEC3 version will be used; for example, `dnssec-keygen -3a RSASHA1` specifies the NSEC3RSASHA1 algorithm.

-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 TKEY, the value must be DH (Diffie Hellman); specifying his value will automatically set the `-T KEY` option as well.

These values are case insensitive. In some cases, abbreviations are supported, such as ECDSA256 for ECDSAP256SHA256 and ECDSA384 for ECDSAP384SHA384. If RSASHA1 is specified along with the `-3` option, then NSEC3RSASHA1 will be used instead.

This parameter *must* be specified except when using the `-S` option, which copies the algorithm from the predecessor key.

In prior releases, HMAC algorithms could be generated for use as TSIG keys, but that feature has been removed as of BIND 9.13.0. Use `tsig-keygen` to generate TSIG keys.

-b keysize Specifies the number of bits in the key. The choice of key size depends on the algorithm used. RSA keys must be between 1024 and 4096 bits. Diffie Hellman keys must be between 128 and 4096 bits. Elliptic curve algorithms don't need this parameter.

If the key size is not specified, some algorithms have pre-defined defaults. For example, RSA keys for use as DNSSEC zone signing keys have a default size of 1024 bits; RSA keys for use as key signing keys (KSKs, generated with `-f KSK`) default to 2048 bits.

-C Compatibility mode: generates an old-style key, without any timing metadata. By default, `dnssec-keygen` will include the key's creation date in the metadata stored with the private key, and other dates may be set there as well (publication date, activation date, etc). Keys that include this data may be incompatible with older versions of BIND; the `-C` option suppresses them.

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

-d bits Key size in bits. For the algorithms RSASHA1, NSEC3RSASA1, RSASHA256 and RSASHA512 the key size must be in range 1024-4096. DH size is between 128 and 4096. This option is ignored for algorithms ECDSAP256SHA256, ECDSAP384SHA384, ED25519 and ED448.

-E engine Specifies the cryptographic hardware to use, when applicable.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-f flag Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flags are KSK (Key Signing Key) and REVOKE.

-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 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.

-h Prints a short summary of the options and arguments to `dnssec-keygen`.

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

-k policy Create keys for a specific dnssec-policy. If a policy uses multiple keys, `dnssec-keygen` will generate multiple keys. This will also create a ".state" file to keep track of the key state.

This option creates keys according to the dnssec-policy configuration, hence it cannot be used together with many of the other options that `dnssec-keygen` provides.

-L ttl 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.

-l file Provide a configuration file that contains a dnssec-policy statement (matching the policy set with `-k`).

-n nametype Specifies the owner type of the key. The value of `nametype` 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.

-p protocol Sets the protocol value for the generated key, for use with `-T 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 `stderr` indicating the progress of the key generation. A '.' indicates that a random number has been found which passed an initial sieve test; '+' 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 key Create a new key which is an explicit successor to an existing key. The name, algorithm, size, and type of the 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 prepublication 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).

-t type Indicates the use of the key, for use with `-T 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 Prints version information.

-v level Sets the debugging level.

11.12.4 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. If not set, and if the -G option has not been used, the default is "now".

-P sync date/offset Sets the date on which CDS and CDNSKEY records that match this key are to be published to the zone.

-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, if and -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.

-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.)

-D sync date/offset Sets the date on which the CDS and CDNSKEY records that match this key are to be deleted.

-i interval Sets the prepublication 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.

11.12.5 Generated Keys

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

- `nnnn` is the key name.
- `aaa` is the numeric representation of the algorithm.
- `iiiii` is the key identifier (or footprint).

`dnssec-keygen` creates two files, with names based on the printed string. `Knnnn.+aaa+iiiii.key` contains the public key, and `Knnnn.+aaa+iiiii.private` contains the private key.

The `.key` file contains a DNSKEY or KEY record. When a zone is being signed by `named` or `dnssec-signzone -S`, DNSKEY records are included automatically. In other cases, the `.key` file can be inserted into a zone file manually or with a `$INCLUDE` statement.

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

11.12.6 Example

To generate an ECDSAP256SHA256 zone-signing key for the zone `example.com`, issue the command:

```
dnssec-keygen -a ECDSAP256SHA256 example.com
```

The command would print a string of the form:

```
Kexample.com.+013+26160
```

In this example, `dnssec-keygen` creates the files `Kexample.com.+013+26160.key` and `Kexample.com.+013+26160.private`.

To generate a matching key-signing key, issue the command:

```
dnssec-keygen -a ECDSAP256SHA256 -f KSK example.com
```

11.12.7 See Also

dnssec-signzone(8), BIND 9 Administrator Reference Manual, **RFC 2539**, **RFC 2845**, **RFC 4034**.

11.13 dnssec-keyfromlabel - DNSSEC key generation tool

11.13.1 Synopsis

dnssec-keyfromlabel {**-l** label} [**-3**] [**-a** algorithm] [**-A** date/offset] [**-c** class] [**-D** date/offset] [**-D** sync date/offset] [**-E** engine] [**-f** flag] [**-G**] [**-I** date/offset] [**-i** interval] [**-k**] [**-K** directory] [**-L** ttl] [**-n** nametype] [**-P** date/offset] [**-P** sync date/offset] [**-p** protocol] [**-R** date/offset] [**-S** key] [**-t** type] [**-v** level] [**-V**] [**-y**] {name}

11.13.2 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`, but the key material is stored within the HSM, and the actual signing takes place there.

The `name` of the key is specified on the command line. This must match the name of the zone for which the key is being generated.

11.13.3 Options

-a algorithm Selects the cryptographic algorithm. The value of `algorithm` must be one of RSASHA1, NSEC3RSASHA1, RSASHA256, RSASHA512, ECDSAP256SHA256, ECDSAP384SHA384, ED25519 or ED448.

If no algorithm is specified, then RSASHA1 will be used by default, unless the `-3` option is specified, in which case NSEC3RSASHA1 will be used instead. (If `-3` is used and an algorithm is specified, that algorithm will be checked for compatibility with NSEC3.)

These values are case insensitive. In some cases, abbreviations are supported, such as ECDSA256 for ECDSAP256SHA256 and ECDSA384 for ECDSAP384SHA384. If RSASHA1 is specified along with the `-3` option, then NSEC3RSASHA1 will be used instead.

As of BIND 9.12.0, this option is mandatory except when using the `-S` option (which copies the algorithm from the predecessory key). Previously, the default for newly generated keys was RSASHA1.

-3 Use an NSEC3-capable algorithm to generate a DNSSEC key. If this option is used with an algorithm that has both NSEC and NSEC3 versions, then the NSEC3 version will be used; for example, `dnssec-keygen -3a RSASHA1` specifies the NSEC3RSASHA1 algorithm.

-E engine Specifies the cryptographic hardware to use.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-l label Specifies the label for a key pair in the crypto hardware.

When BIND 9 is built with OpenSSL-based PKCS#11 support, the label is an arbitrary string that identifies a particular key. It may be preceded by an optional OpenSSL engine name, followed by a colon, as in "pkcs11:keylabel".

When BIND 9 is built with native PKCS#11 support, the label is a PKCS#11 URI string in the format "pkcs11:`keyword`=value[;`keyword`=value;…]" Keywords include "token", which identifies the HSM; "object", which identifies the key; and "pin-source", which identifies a file from which the HSM's PIN code can be obtained. The label will be stored in the on-disk "private" file.

If the label 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. Note: 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 nametype Specifies the owner type of the key. The value of `nametype` 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 Compatibility mode: generates an old-style key, without any metadata. By default, `dnssec-keyfromlabel` will include the key's creation date in the metadata stored with the private key, and other dates may be set there as well (publication date, activation date, etc). Keys that include this data may be incompatible with older versions of BIND; the `-C` option suppresses them.

-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 (Key Signing Key) and REVOKE.

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

-h Prints a short summary of the options and arguments to `dnssec-keyfromlabel`.

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

-k Generate KEY records rather than DNSKEY records.

-L ttl 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.

-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 key Generate a key as an explicit successor to an existing key. The name, algorithm, size, and type of the key will be set to match the predecessor. 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 prepublication interval, which defaults to 30 days.

-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 Sets the debugging level.

-V Prints version information.

-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. (This is only safe to use if you are sure you won't be using **RFC 5011** trust anchor maintenance with either of the keys involved.)

11.13.4 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".

-P sync date/offset Sets the date on which the CDS and CDNSKEY records which match this key are to be published to the zone.

-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".

-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.

-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.)

-D sync date/offset Sets the date on which the CDS and CDNSKEY records which match this key are to be deleted.

-i interval Sets the prepublication 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.

11.13.5 Generated Key Files

When `dnssec-keyfromlabel` completes successfully, it prints a string of the form `Knnnn.+aaa+iiiii` 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.
- `iiiii` is the key identifier (or footprint).

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

The `.key` file contains a DNS KEY 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.

11.13.6 See Also

dnssec-keygen(8), *dnssec-signzone(8)*, BIND 9 Administrator Reference Manual, **RFC 4034**, The PKCS#11 URI Scheme (draft-pechanec-pkcs11uri-13).

11.14 dnssec-verify - DNSSEC zone verification tool

11.14.1 Synopsis

dnssec-verify [**-c** class] [**-E** engine] [**-I** input-format] [**-o** origin] [**-q**] [**-v** level] [**-V**] [**-x**] [**-z**] {zonefile}

11.14.2 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.

11.14.3 Options

-c class Specifies the DNS class of the zone.

-E engine Specifies the cryptographic hardware to use, when applicable.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-I input-format The format of the input zone file. Possible formats are `"text"` (default) and `"raw"`. This option is primarily intended to be used for dynamic signed zones so that the dumped zone file in a non-text format containing updates can be verified independently. The use of this option does not make much sense for non-dynamic zones.

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

-v level Sets the debugging level.

-V Prints version information.

-q Quiet mode: Suppresses output. Without this option, when `dnssec-verify` is run it will print to standard output the number of keys in use, the algorithms used to verify the zone was signed correctly and other status information. With it, all non-error output is suppressed, and only the exit code will indicate success.

-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 in `dnssec-signzone`.

-z Ignore the KSK flag on the keys when determining whether the zone if 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 in `dnssec-signzone`.

zonefile The file containing the zone to be signed.

11.14.4 See Also

`dnssec-signzone(8)`, BIND 9 Administrator Reference Manual, **RFC 4033**.

11.15 dnssec-settime: set the key timing metadata for a DNSSEC key

11.15.1 Synopsis

`dnssec-settime` [**-f**] [**-K** directory] [**-L** ttl] [**-P** date/offset] [**-P** sync date/offset] [**-A** date/offset] [**-R** date/offset] [**-I** date/offset] [**-D** date/offset] [**-D** sync date/offset] [**-S** key] [**-i** interval] [**-h**] [**-V**] [**-v** level] [**-E** engine] {keyfile} [**-s**] [**-g** state] [**-d** state date/offset] [**-k** state date/offset] [**-r** state date/offset] [**-z** state date/offset]

11.15.2 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` 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+iiiii.key` and `Knnnn.+aaa+iiiii.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).

When working with state files, it is possible to update the timing metadata in those files as well with `-s`. If this option is used you can also update key states with `-d` (DS), `-k` (DNSKEY), `-r` (RRSIG of KSK), or `-z` (RRSIG of ZSK). Allowed states are HIDDEN, RUMOURED, OMNIPRESENT, and UNRETENTIVE.

You can also set the goal state of the key with `-g`. This should be either HIDDEN or OMNIPRESENT (representing whether the key should be removed from the zone, or published).

It is NOT RECOMMENDED to manipulate state files manually except for testing purposes.

11.15.3 Options

-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.

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

-L ttl 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 Emit usage message and exit.

-V Prints version information.

-v level Sets the debugging level.

-E engine Specifies the cryptographic hardware to use, when applicable.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

11.15.4 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 unset a date, 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.

-P sync date/offset Sets the date on which CDS and CDNSKEY records that match this key are to be published to the zone.

-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.

-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.

-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.)

-D sync date/offset Sets the date on which the CDS and CDNSKEY records that match this key are to be deleted.

-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.

-i interval Sets the prepublication 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 set to be 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.

11.15.5 Key State Options

Known key states are HIDDEN, RUMOURED, OMNIPRESENT and UNRETENTIVE. These should not be set manually except for testing purposes.

-s When setting key timing data, also update the state file.

-g Set the goal state for this key. Must be HIDDEN or OMNIPRESENT.

-d Set the DS state for this key, and when it was last changed.

-k Set the DNSKEY state for this key, and when it was last changed.

-r Set the RRSIG (KSK) state for this key, and when it was last changed.

-z

Set the RRSIG (ZSK) state for this key, and when it was last changed.

11.15.6 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/Psync/A/R/I/D/Dsync/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 or strings to indicate which value or values to print: `C` for the creation date, `P` for the publication date, `Psync` for the CDS and CDNSKEY publication date, `A` for the activation date, `R` for the revocation date, `I` for the inactivation date, `D` for the deletion date, and `Dsync` for the CDS and CDNSKEY deletion date To print all of the metadata, use `-p all`.

11.15.7 See Also

dnssec-keygen(8), *dnssec-signzone(8)*, BIND 9 Administrator Reference Manual, **RFC 5011**.

11.16 dnssec-importkey - import DNSKEY records from external systems so they can be managed

11.16.1 Synopsis

dnssec-importkey [**-K** directory] [**-L** ttl] [**-P** date/offset] [**-P** sync date/offset] [**-D** date/offset] [**-D** sync date/offset] [**-h**] [**-v** level] [**-V**] {keyfile}

dnssec-importkey {**-f** filename} [**-K** directory] [**-L** ttl] [**-P** date/offset] [**-P** sync date/offset] [**-D** date/offset] [**-D** sync date/offset] [**-h**] [**-v** level] [**-V**] [dnsname]

11.16.2 Description

`dnssec-importkey` reads a public DNSKEY record and generates a pair of .key/.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.

11.16.3 Options

-f filename 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 ttl 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 Emit usage message and exit.

-v level Sets the debugging level.

-V Prints version information.

11.16.4 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.

-P sync date/offset Sets the date on which CDS and CDNSKEY records that match this key are to be published to the zone.

-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.)

-D sync date/offset Sets the date on which the CDS and CDNSKEY records that match this key are to be deleted.

11.16.5 Files

A keyfile can be designed by the key identification `Knnnn.+aaa+iiiii` or the full file name `Knnnn.+aaa+iiiii.key` as generated by dnssec-keygen8.

11.16.6 See Also

dnssec-keygen(8), *dnssec-signzone(8)*, BIND 9 Administrator Reference Manual, **RFC 5011**.

11.17 dnssec-signzone - DNSSEC zone signing tool

11.17.1 Synopsis

`dnssec-signzone` [**-a**] [**-c** class] [**-d** directory] [**-D**] [**-E** engine] [**-e** end-time] [**-f** output-file] [**-g**] [**-h**] [**-i** interval] [**-I** input-format] [**-j** jitter] [**-K** directory] [**-k** key] [**-L** serial] [**-M** maxttl] [**-N** soa-serial-format] [**-o** origin] [**-O** output-format] [**-P**] [**-Q**] [**-q**] [**-R**] [**-S**] [**-s** start-time] [**-T** ttl] [**-t**] [**-u**] [**-v** level] [**-V**] [**-X** extended end-time] [**-x**] [**-z**] [**-3** salt] [**-H** iterations] [**-A**] {zonefile} [key…]

11.17.2 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.

11.17.3 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 `$INCLUDE`. This option cannot be combined with `-O raw`, `-O map`, or serial number updating.

-E engine When applicable, specifies the hardware to use for cryptographic operations, such as a secure key store used for signing.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-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.

-k key Treat specified key as a key signing key ignoring any key flags. This option may be specified multiple times.

-M maxttl 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` in `named.conf`. (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 no `start-time` is 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 `start-time`, 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 no `end-time` is specified, 30 days from the start time is used as a default. `end-time` must be later than `start-time`.

-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 `start-time`, 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 no `extended end-time` is specified, the value of `end-time` is used as the default. (`end-time`, in turn, defaults to 30 days from the start time.) `extended end-time` must be later than `start-time`.

-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 the standard output, with a default output format of "full".

-h Prints a short summary of the options and arguments to `dnssec-signzone`.

-V Prints version information.

-i interval When a previously-signed zone is passed as input, records may be resigned. The `interval` option specifies the cycle interval as an offset from the current time (in seconds). If a 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 `end-time` or `start-time` 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.

-I input-format The format of the input zone file. Possible formats are `"text"` (default), `"raw"`, and `"map"`. This option is primarily intended to be used for dynamic signed zones so that the dumped zone file in a non-text format containing updates can be signed directly. The use of this option does not make much sense for non-dynamic zones.

-j jitter When signing a zone with a fixed signature lifetime, all RRSIG records issued at the time of signing expires 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` option 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 to "raw" or "map" format, set the "source serial" value in the header to the specified serial number. (This is expected to be used primarily for testing purposes.)

-n ncpus Specifies the number of threads to use. By default, one thread is started for each detected CPU.

-N soa-serial-format The SOA serial number format of the signed zone. Possible formats are `"keep"` (default), `"increment"`, `"unixtime"`, and `"date"`.

`"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.

`"date"` Set the SOA serial number to today's date in YYYYMMDDNN format.

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

-O output-format The format of the output file containing the signed zone. Possible formats are `"text"` (default), which is the standard textual representation of the zone; `"full"`, which is text output in a format suitable for processing by external scripts; and `"map"`, `"raw"`, and `"raw=N"`, which store the zone in binary formats for rapid loading by `named`. `"raw=N"` specifies the format version of the raw zone file: if N is 0, the raw file can be read by any version of `named`; if N is 1, the file can be read by release 9.9.0 or higher; the default is 1.

-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` forces `dnssec-signzone` to remove signatures from keys that are no longer active. This enables ZSK rollover using the procedure described in **RFC 4641#4.2.1.1** ("Pre-Publish Key Rollover").

-q Quiet mode: Suppresses unnecessary output. Without this option, when `dnssec-signzone` is run it will print to standard output the number of keys in use, the algorithms used to verify the zone was signed correctly and other status information, and finally the filename containing the signed zone. With it, that output is suppressed, leaving only the filename.

-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 4641#4.2.1.2** ("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:

If no timing metadata has been set for the key, the key is published in the zone and used to sign the zone.

If the key's publication date is set and is in the past, the key is published in the zone.

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.

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.

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.

If key's sync publication date is set and in the past, synchronization records (type CDS and/or CDNSKEY) are created.

If key's sync deletion date is set and in the past, synchronization records (type CDS and/or CDNSKEY) are removed.

-T ttl 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 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.

-v level Sets the debugging level.

-x Only sign the DNSKEY, CDNSKEY, and CDS RRsets with key-signing keys, and omit signatures from zone-signing keys. (This is similar to the `dnssec-dnskey-kskonly yes;` zone option in `named`.)

-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 no;` zone option in `named`.)

-3 salt Generate an NSEC3 chain with the given hex encoded salt. A dash (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.

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.

11.17.4 Example

The following command signs the `example.com` zone with the ECDSAP256SHA256 key generated by key generated by `dnssec-keygen` (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` file.

This 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
%
```

11.17.5 See Also

dnssec-keygen(8), BIND 9 Administrator Reference Manual, **RFC 4033**, **RFC 4641**.

11.18 dnssec-dsfromkey - DNSSEC DS RR generation tool

11.18.1 Synopsis

dnssec-dsfromkey [**-1** | **-2** | **-a** alg] [**-C**] [**-T** TTL] [**-v** level] [**-K** directory] {keyfile}

dnssec-dsfromkey [**-1** | **-2** | **-a** alg] [**-C**] [**-T** TTL] [**-v** level] [**-c** class] [**-A**] {**-f** file} [dnsname]

dnssec-dsfromkey [**-1** | **-2** | **-a** alg] [**-C**] [**-T** TTL] [**-v** level] [**-c** class] [**-K** directory] {**-s**} {dnsname}

dnssec-dsfromkey [**-h** | **-V**]

11.18.2 Description

The `dnssec-dsfromkey` command outputs DS (Delegation Signer) resource records (RRs), or CDS (Child DS) RRs with the `-C` option.

The input keys can be specified in a number of ways:

By default, `dnssec-dsfromkey` reads a key file named like `Knnnn.+aaa+iiiii.key`, as generated by `dnssec-keygen`.

With the `-f file` option, `dnssec-dsfromkey` reads keys from a zone file or partial zone file (which can contain just the DNSKEY records).

With the `-s` option, `dnssec-dsfromkey` reads a `keyset-` file, as generated by `dnssec-keygen -C`.

11.18.3 Options

-1 An abbreviation for `-a SHA1`

-2 An abbreviation for `-a SHA-256`

-a algorithm Specify a digest algorithm to use when converting DNSKEY records to DS records. This option can be repeated, so that multiple DS records are created for each DNSKEY record.

The algorithm must be one of SHA-1, SHA-256, or SHA-384. These values are case insensitive, and the hyphen may be omitted. If no algorithm is specified, the default is SHA-256.

-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 `-f` zone file mode.

-c class Specifies the DNS class (default is IN). Useful only in `-s` keyset or `-f` zone file mode.

-C Generate CDS records rather than DS records.

-f file Zone file mode: `dnssec-dsfromkey`'s final dnsname argument is the DNS domain name of a zone whose master file can be read from `file`. If the zone name is the same as `file`, then it may be omitted.

If file is `"-"`, then the zone data is read from the standard input. This makes it possible to use the output of the `dig` command as input, as in:

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

-h Prints usage information.

-K directory Look for key files or `keyset-` files in `directory`.

-s Keyset mode: `dnssec-dsfromkey`'s final dnsname argument is the DNS domain name used to locate a `keyset-` file.

-T TTL Specifies the TTL of the DS records. By default the TTL is omitted.

-v level Sets the debugging level.

-V Prints version information.

11.18.4 Example

To build the SHA-256 DS RR from the `Kexample.com.+003+26160` keyfile name, you can issue the following command:

```
dnssec-dsfromkey -2 Kexample.com.+003+26160
```

The command would print something like:

```
example.com. IN DS 26160 5 2 3A1EADA7A74B8D0BA86726B0C227AA85AB8BBD2B2004F41A868A54F0C5EA0B94
```

11.18.5 Files

The keyfile can be designated by the key identification `Knnnn.+aaa+iiiii` or the full file name `Knnnn.+aaa+iiiii.key` as generated by dnssec-keygen8.

The keyset file name is built from the `directory`, the string `keyset-` and the `dnsname`.

11.18.6 Caveat

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

11.18.7 See Also

dnssec-keygen(8), *dnssec-signzone(8)*, BIND 9 Administrator Reference Manual, **RFC 3658** (DS RRs), **RFC 4509** (SHA-256 for DS RRs), **RFC 6605** (SHA-384 for DS RRs), **RFC 7344** (CDS and CDNSKEY RRs).

11.19 dnssec-checkds - DNSSEC delegation consistency checking tool

11.19.1 Synopsis

`dnssec-checkds` [**-d***dig path*] [**-D***dsfromkey path*] [**-f***file*] [**-l***domain*] [**-s***file*] {zone}

11.19.2 Description

`dnssec-checkds` verifies the correctness of Delegation Signer (DS) resource records for keys in a specified zone.

11.19.3 Options

-a *algorithm*

> Specify a digest algorithm to use when converting the zones DNSKEY records to expected DS records. This option can be repeated, so that multiple records are checked for each DNSKEY record.
>
> The *algorithm* must be one of SHA-1, SHA-256, or SHA-384. These values are case insensitive, and the hyphen may be omitted. If no algorithm is specified, the default is SHA-256.

-f *file*

> If a `file` is specified, then the zone is read from that file to find the DNSKEY records. If not, then the DNSKEY records for the zone are looked up in the DNS.

-s *file*

> Specifies a prepared dsset file, such as would be generated by `dnssec-signzone`, to use as a source for the DS RRset instead of querying the parent.

-d *dig path*

Specifies a path to a `dig` binary. Used for testing.

-D *dsfromkey path*

Specifies a path to a `dnssec-dsfromkey` binary. Used for testing.

11.19.4 See Also

`dnssec-dsfromkey`(8), `dnssec-keygen`(8), `dnssec-signzone`(8),

11.20 dnssec-coverage - checks future DNSKEY coverage for a zone

11.20.1 Synopsis

`dnssec-coverage` [**-K***directory*] [**-l***length*] [**-f***file*] [**-d***DNSKEY TTL*] [**-m***max TTL*] [**-r***interval*] [**-c***compilezone path*] [**-k**] [**-z**] [zone...]

11.20.2 Description

`dnssec-coverage` verifies that the DNSSEC keys for a given zone or a set of zones have timing metadata set properly to ensure no future lapses in DNSSEC coverage.

If `zone` is specified, then keys found in the key repository matching that zone are scanned, and an ordered list is generated of the events scheduled for that key (i.e., publication, activation, inactivation, deletion). The list of events is walked in order of occurrence. Warnings are generated if any event is scheduled which could cause the zone to enter a state in which validation failures might occur: for example, if the number of published or active keys for a given algorithm drops to zero, or if a key is deleted from the zone too soon after a new key is rolled, and cached data signed by the prior key has not had time to expire from resolver caches.

If `zone` is not specified, then all keys in the key repository will be scanned, and all zones for which there are keys will be analyzed. (Note: This method of reporting is only accurate if all the zones that have keys in a given repository share the same TTL parameters.)

11.20.3 Options

-K *directory*

Sets the directory in which keys can be found. Defaults to the current working directory.

-f *file*

If a `file` is specified, then the zone is read from that file; the largest TTL and the DNSKEY TTL are determined directly from the zone data, and the `-m` and `-d` options do not need to be specified on the command line.

-l *duration*

The length of time to check for DNSSEC coverage. Key events scheduled further into the future than `duration` will be ignored, and assumed to be correct.

The value of `duration` can be set in seconds, or in larger units of time by adding a suffix: mi for minutes, h for hours, d for days, w for weeks, mo for months, y for years.

-m *maximum TTL*

Sets the value to be used as the maximum TTL for the zone or zones being analyzed when determining whether there is a possibility of validation failure. When a zone-signing key is deactivated, there must be enough time for the record in the zone with the longest TTL to have expired from resolver caches before that key can be purged from the DNSKEY RRset. If that condition does not apply, a warning will be generated.

The length of the TTL can be set in seconds, or in larger units of time by adding a suffix: mi for minutes, h for hours, d for days, w for weeks, mo for months, y for years.

This option is not necessary if the `-f` has been used to specify a zone file. If `-f` has been specified, this option may still be used; it will override the value found in the file.

If this option is not used and the maximum TTL cannot be retrieved from a zone file, a warning is generated and a default value of 1 week is used.

-d *DNSKEY TTL*

Sets the value to be used as the DNSKEY TTL for the zone or zones being analyzed when determining whether there is a possibility of validation failure. When a key is rolled (that is, replaced with a new key), there must be enough time for the old DNSKEY RRset to have expired from resolver caches before the new key is activated and begins generating signatures. If that condition does not apply, a warning will be generated.

The length of the TTL can be set in seconds, or in larger units of time by adding a suffix: mi for minutes, h for hours, d for days, w for weeks, mo for months, y for years.

This option is not necessary if `-f` has been used to specify a zone file from which the TTL of the DNSKEY RRset can be read, or if a default key TTL was set using ith the `-L` to `dnssec-keygen`. If either of those is true, this option may still be used; it will override the values found in the zone file or the key file.

If this option is not used and the key TTL cannot be retrieved from the zone file or the key file, then a warning is generated and a default value of 1 day is used.

-r *resign interval*

Sets the value to be used as the resign interval for the zone or zones being analyzed when determining whether there is a possibility of validation failure. This value defaults to 22.5 days, which is also the default in `named`. However, if it has been changed by the `sig-validity-interval` option in named.conf, then it should also be changed here.

The length of the interval can be set in seconds, or in larger units of time by adding a suffix: mi for minutes, h for hours, d for days, w for weeks, mo for months, y for years.

-k

Only check KSK coverage; ignore ZSK events. Cannot be used with `-z`.

-z

Only check ZSK coverage; ignore KSK events. Cannot be used with `-k`.

-c *compilezone path*

Specifies a path to a `named-compilezone` binary. Used for testing.

11.20.4 See Also

`dnssec-checkds`(8), `dnssec-dsfromkey`(8), `dnssec-keygen`(8), `dnssec-signzone`(8)

11.21 dnssec-keymgr - Ensures correct DNSKEY coverage based on a defined policy

11.21.1 Synopsis

:program:`dnssec-keymgr` [**-K***directory*] [**-c***file*] [**-f**] [**-k**] [**-q**] [**-v**] [**-z**] [**-g***path*] [**-s***path*] [zone...]

11.21.2 Description

`dnssec-keymgr` is a high level Python wrapper to facilitate the key rollover process for zones handled by BIND. It uses the BIND commands for manipulating DNSSEC key metadata: `dnssec-keygen` and `dnssec-settime`.

DNSSEC policy can be read from a configuration file (default /etc/dnssec-policy.conf), from which the key parameters, publication and rollover schedule, and desired coverage duration for any given zone can be determined. This file may be used to define individual DNSSEC policies on a per-zone basis, or to set a "default" policy used for all zones.

When `dnssec-keymgr` runs, it examines the DNSSEC keys for one or more zones, comparing their timing metadata against the policies for those zones. If key settings do not conform to the DNSSEC policy (for example, because the policy has been changed), they are automatically corrected.

A zone policy can specify a duration for which we want to ensure the key correctness (`coverage`). It can also specify a rollover period (`roll-period`). If policy indicates that a key should roll over before the coverage period ends, then a successor key will automatically be created and added to the end of the key series.

If zones are specified on the command line, `dnssec-keymgr` will examine only those zones. If a specified zone does not already have keys in place, then keys will be generated for it according to policy.

If zones are *not* specified on the command line, then `dnssec-keymgr` will search the key directory (either the current working directory or the directory set by the `-K` option), and check the keys for all the zones represented in the directory.

Key times that are in the past will not be updated unless the `-f` is used (see below). Key inactivation and deletion times that are less than five minutes in the future will be delayed by five minutes.

It is expected that this tool will be run automatically and unattended (for example, by `cron`).

11.21.3 Options

-c *file*

> If `-c` is specified, then the DNSSEC policy is read from `file`. (If not specified, then the policy is read from /etc/dnssec-policy.conf; if that file doesnt exist, a built-in global default policy is used.)

-f

> Force: allow updating of key events even if they are already in the past. This is not recommended for use with zones in which keys have already been published. However, if a set of keys has been generated all of which have publication and activation dates in the past, but the keys have not been published in a zone as yet, then this option can be used to clean them up and turn them into a proper series of keys with appropriate rollover intervals.

-g *keygen-path*

> Specifies a path to a `dnssec-keygen` binary. Used for testing. See also the `-s` option.

-h

Print the `dnssec-keymgr` help summary and exit.

-K *directory*

Sets the directory in which keys can be found. Defaults to the current working directory.

-k

Only apply policies to KSK keys. See also the `-z` option.

-q

Quiet: suppress printing of `dnssec-keygen` and `dnssec-settime`.

-s *settime-path*

Specifies a path to a `dnssec-settime` binary. Used for testing. See also the `-g` option.

-v

Print the `dnssec-keymgr` version and exit.

-z

Only apply policies to ZSK keys. See also the `-k` option.

11.21.4 Policy Configuration

The dnssec-policy.conf file can specify three kinds of policies:

· *Policy classes* (`policy`*name*`{ ... };`) can be inherited by zone policies or other policy classes; these can be used to create sets of different security profiles. For example, a policy class `normal` might specify 1024-bit key sizes, but a class `extra` might specify 2048 bits instead; `extra` would be used for zones that had unusually high security needs.

· *Algorithm policies:* (`algorithm-policy`*algorithm*`{ ... };`) override default per-algorithm settings. For example, by default, RSASHA256 keys use 2048-bit key sizes for both KSK and ZSK. This can be modified using `algorithm-policy`, and the new key sizes would then be used for any key of type RSASHA256.

· *Zone policies:* (`zone`*name*`{ ... };`) set policy for a single zone by name. A zone policy can inherit a policy class by including a `policy` option. Zone names beginning with digits (i.e., 0-9) must be quoted. If a zone does not have its own policy then the "default" policy applies.

Options that can be specified in policies:

`algorithm` *name*;

The key algorithm. If no policy is defined, the default is RSASHA256.

`coverage` *duration*;

The length of time to ensure that keys will be correct; no action will be taken to create new keys to be activated after this time. This can be represented as a number of seconds, or as a duration using human-readable units (examples: "1y" or "6 months"). A default value for this option can be set in algorithm policies as well as in policy classes or zone policies. If no policy is configured, the default is six months.

`directory` *path*;

Specifies the directory in which keys should be stored.

`key-size` *keytype size*;

Specifies the number of bits to use in creating keys. The keytype is either "zsk" or "ksk". A default value for this option can be set in algorithm policies as well as in policy classes or zone policies. If no policy is configured, the default is 2048 bits for RSA keys.

`keyttl` *duration*;

The key TTL. If no policy is defined, the default is one hour.

`post-publish` *keytype duration*;

How long after inactivation a key should be deleted from the zone. Note: If `roll-period` is not set, this value is ignored. The keytype is either "zsk" or "ksk". A default duration for this option can be set in algorithm policies as well as in policy classes or zone policies. The default is one month.

`pre-publish` *keytype duration*;

How long before activation a key should be published. Note: If `roll-period` is not set, this value is ignored. The keytype is either "zsk" or "ksk". A default duration for this option can be set in algorithm policies as well as in policy classes or zone policies. The default is one month.

`roll-period` *keytype duration*;

How frequently keys should be rolled over. The keytype is either "zsk" or "ksk". A default duration for this option can be set in algorithm policies as well as in policy classes or zone policies. If no policy is configured, the default is one year for ZSKs. KSKs do not roll over by default.

`standby` *keytype number*;

Not yet implemented.

11.21.5 Remaining Work

· Enable scheduling of KSK rollovers using the `-P sync` and `-D sync` options to `dnssec-keygen` and `dnssec-settime`. Check the parent zone (as in `dnssec-checkds`) to determine when its safe for the key to roll.

· Allow configuration of standby keys and use of the REVOKE bit, for keys that use RFC 5011 semantics.

11.21.6 See Also

`dnssec-coverage`(8), `dnssec-keygen`(8), `dnssec-settime`(8), `dnssec-checkds`(8)

11.22 filter-aaaa.so - filter AAAA in DNS responses when A is present

11.22.1 Synopsis

`plugin query` "filter-aaaa.so" [{ parameters }];

11.22.2 Description

`filter-aaaa.so` is a query plugin module for `named`, enabling `named` to omit some IPv6 addresses when responding to clients.

Until BIND 9.12, this feature was implemented natively in `named` and enabled with the `filter-aaaa` ACL and the `filter-aaaa-on-v4` and `filter-aaaa-on-v6` options. These options are now deprecated in `named.conf`, but can be passed as parameters to the `filter-aaaa.so` plugin, for example:

```
plugin query "/usr/local/lib/filter-aaaa.so" {
        filter-aaaa-on-v4 yes;
        filter-aaaa-on-v6 yes;
        filter-aaaa { 192.0.2.1; 2001:db8:2::1; };
};
```

This module is intended to aid transition from IPv4 to IPv6 by withholding IPv6 addresses from DNS clients which are not connected to the IPv6 Internet, when the name being looked up has an IPv4 address available. Use of this module is not recommended unless absolutely necessary.

Note: This mechanism can erroneously cause other servers not to give AAAA records to their clients. If a recursing server with both IPv6 and IPv4 network connections queries an authoritative server using this mechanism via IPv4, it will be denied AAAA records even if its client is using IPv6.

11.22.3 Options

`filter-aaaa` Specifies a list of client addresses for which AAAA filtering is to be applied. The default is `any`.

`filter-aaaa-on-v4` If set to `yes`, the DNS client is at an IPv4 address, in `filter-aaaa`, and if the response does not include DNSSEC signatures, then all AAAA records are deleted from the response. This filtering applies to all responses and not only authoritative responses.

If set to `break-dnssec`, then AAAA records are deleted even when DNSSEC is enabled. As suggested by the name, this causes the response to fail to verify, because the DNSSEC protocol is designed to detect deletions.

This mechanism can erroneously cause other servers not to give AAAA records to their clients. A recursing server with both IPv6 and IPv4 network connections that queries an authoritative server using this mechanism via IPv4 will be denied AAAA records even if its client is using IPv6.

`filter-aaaa-on-v6` Identical to `filter-aaaa-on-v4`, except it filters AAAA responses to queries from IPv6 clients instead of IPv4 clients. To filter all responses, set both options to `yes`.

11.22.4 See Also

BIND 9 Administrator Reference Manual.

11.23 ddns-confgen - ddns key generation tool

11.23.1 Synopsis

tsig-keygen [**-a** algorithm] [**-h**] [**-r** randomfile] [**-s** name]

ddns-confgen [**-a** algorithm] [**-h**] [**-k** keyname] [**-q**] [**-r** randomfile] [**-s** name] [**-z** zone]

11.23.2 Description

`tsig-keygen` and `ddns-confgen` are invocation methods for a utility that generates keys for use in TSIG signing. The resulting keys can be used, for example, to secure dynamic DNS updates to a zone or for the `rndc` command channel.

When run as `tsig-keygen`, a domain name can be specified on the command line which will be used as the name of the generated key. If no name is specified, the default is `tsig-key`.

When run as `ddns-confgen`, 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 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.

11.23.3 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 Prints a short summary of options and arguments.

-k keyname Specifies the key name of the DDNS authentication key. The default is `ddns-key` when neither the `-s` nor `-z` option is 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 (`ddns-confgen` only.) Quiet mode: Print only the key, with no explanatory text or usage examples; This is essentially identical to `tsig-keygen`.

-s name (`ddns-confgen` only.) Generate configuration example to allow dynamic updates of a single hostname. The example `named.conf` text 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.

-z zone (`ddns-confgen` only.) Generate configuration example to allow dynamic updates of a zone: The example `named.conf` text 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.

11.23.4 See Also

nsupdate(1), *named.conf(5)*, *named(8)*, BIND 9 Administrator Reference Manual.

11.24 rndc-confgen - rndc key generation tool

11.24.1 Synopsis

`rndc-confgen` [**-a**] [**-A** algorithm] [**-b** keysize] [**-c** keyfile] [**-h**] [**-k** keyname] [**-p** port] [**-s** address] [**-t** chrootdir] [**-u** user]

11.24.2 Description

`rndc-confgen` generates configuration files for `rndc`. It can be used as a convenient alternative to writing the `rndc.conf` file and the corresponding `controls` and `key` statements in `named.conf` 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.

11.24.3 Arguments

-a Do automatic `rndc` configuration. This creates a file `rndc.key` in `/etc` (or whatever `sysconfdir` was specified as when BIND was built) that is read by both `rndc` and `named` on startup. The `rndc.key` file defines a default command channel and authentication key allowing `rndc` to communicate with `named` on the local host with no further configuration.

Running `rndc-confgen -a` allows BIND 9 and `rndc` to be used as drop-in replacements for BIND 8 and `ndc`, with no changes to the existing BIND 8 `named.conf` file.

If a more elaborate configuration than that generated by `rndc-confgen -a` is required, for example if rndc 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 TSIG key. Available choices are: hmac-md5, hmac-sha1, hmac-sha224, hmac-sha256, hmac-sha384 and hmac-sha512. The default is hmac-sha256.

-b keysize Specifies the size of the authentication key in bits. Must be between 1 and 512 bits; the default is the hash size.

-c keyfile Used with the `-a` option to specify an alternate location for `rndc.key`.

-h Prints a short summary of the options and arguments to `rndc-confgen`.

-k keyname 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` listens for connections from `rndc`. The default is 953.

-s address Specifies the IP address where `named` listens for command channel connections from `rndc`. The default is the loopback address 127.0.0.1.

-t chrootdir Used with the `-a` option to specify a directory where `named` will run chrooted. An additional copy of the `rndc.key` will be written relative to this directory so that it will be found by the chrooted `named`.

-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 area has its owner changed.

11.24.4 Examples

To allow `rndc` 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`, run

```
rndc-confgen
```

11.24.5 See Also

rndc(8), *rndc.conf(5)*, *named(8)*, BIND 9 Administrator Reference Manual.

11.25 delv - DNS lookup and validation utility

11.25.1 Synopsis

delv [@server] [[**-4**] | [**-6**]] [**-a** anchor-file] [**-b** address] [**-c** class] [**-d** level] [**-i**] [**-m**] [**-p** port#] [**-q** name] [**-t** type] [**-x** addr] [name] [type] [class] [queryopt...]

delv [**-h**]

delv [**-v**]

delv [queryopt...] [query...]

11.25.2 Description

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

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 server to carry out validation, it can be used to check the validity of DNS responses in environments where local name servers 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).

11.25.3 Simple Usage

A typical invocation of **delv** looks like:

```
delv @server name type
```

where:

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 name server (note, however, that this initial lookup is *not* validated by DNSSEC).

If no `server` argument is provided, `delv` consults `/etc/resolv.conf`; if an address is found there, it queries the name server at that address. If either of the `-4` or `-6` options are in use, then only addresses for the corresponding transport 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).

`name` is the domain name to be looked up.

`type` indicates what type of query is required MDASH ANY, A, MX, etc. `type` can be any valid query type. If no `type` argument is supplied, `delv` will perform a lookup for an A record.

11.25.4 Options

-a anchor-file Specifies a file from which to read DNSSEC trust anchors. The default is `/etc/bind.keys`, which is included with BIND 9 and contains one or more trust anchors for the root zone (".").

Keys that do not match the root zone name are ignored. An alternate key name can be specified using the `+root=NAME` options.

Note: When reading the trust anchor file, `delv` treat `trust-anchors initial-key` and `static-key` identically. That is, for a managed key, it is the *initial* key that is trusted; **RFC 5011** key management is not supported. `delv` will not consult the managed-keys database maintained by `named`. This means that if either of the keys in `/etc/bind.keys` is revoked and rolled over, it will be necessary to update `/etc/bind.keys` to use DNSSEC validation in `delv`.

-b address 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.

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

-h Display the `delv` help usage output and exit.

-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`.)

-m Enables memory usage debugging.

-p port# Specifies a destination port to use for queries instead of 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 query type to `type`, which can be any valid query type supported in BIND 9 except for zone transfer types AXFR and IXFR. As with `-q`, this 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", unless the `-x` option is supplied to indicate a reverse lookup, in which case it is "PTR".

-v Print the `delv` version and exit.

-x addr Performs a reverse lookup, mapping an addresses to a name. `addr` 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` or `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.

11.25.5 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:

+[no]cdflag 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.

+[no]class Controls whether to display the CLASS when printing a record. The default is to display the CLASS.

+[no]ttl Controls whether to display the TTL when printing a record. The default is to display the TTL.

+[no]rtrace 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 systemwide debug level to 1 using the `-d` option will product the same output (but will affect other logging categories as well).

+[no]mtrace 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 systemwide debug level to 10 using the `-d` option will produce the same output (but will affect other logging categories as well).

+[no]vtrace 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 systemwide debug level to 3 using the `-d` option will produce the same output (but will affect other logging categories as well).

+[no]short Provide a terse answer. The default is to print the answer in a verbose form.

+[no]comments Toggle the display of comment lines in the output. The default is to print comments.

+[no]rrcomments 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.

+[no]crypto 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]".

+[no]trust Controls whether to display the trust level when printing a record. The default is to display the trust level.

+[no]split[=W] 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 multiline mode is active.

+[no]all Set or clear the display options `+[no]comments`, `+[no]rrcomments`, and `+[no]trust` as a group.

+[no]multiline 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.

+[no]dnssec Indicates whether to display RRSIG records in the `delv` output. The default is to do so. Note that (unlike in `dig`) 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`.

+[no]root[=ROOT] Indicates whether to perform conventional 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.

+[no]tcp Controls whether to use TCP when sending queries. The default is to use UDP unless a truncated response has been received.

+[no]unknownformat Print all RDATA in unknown RR type presentation format (**RFC 3597**). The default is to print RDATA for known types in the type's presentation format.

+[no]yaml Print response data in YAML format.

11.25.6 Files

`/etc/bind.keys`

`/etc/resolv.conf`

11.25.7 See Also

dig(1), *named(8)*, **RFC 4034**, **RFC 4035**, **RFC 4431**, **RFC 5074**, **RFC 5155**.

11.26 nsupdate - dynamic DNS update utility

11.26.1 Synopsis

nsupdate [**-d**] [**-D**] [**-i**] [**-L** level] [[**-g**] | [**-o**] | [**-l**] | [**-y** [hmac:]keyname:secret] | [**-k** keyfile]] [**-t** timeout] [**-u** udptimeout] [**-r** udpretries] [**-v**] [**-T**] [**-P**] [**-V**] [[**-4**] | [**-6**]] [filename]

11.26.2 Description

`nsupdate` is used to submit Dynamic DNS Update requests as defined in **RFC 2136** to a name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control via `nsupdate` or a DHCP server 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 Dynamic DNS updates. These use the TSIG resource record type described in **RFC 2845** or the SIG(0) record described in **RFC 2535** and **RFC 2931** or GSS-TSIG as described in **RFC 3645**.

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 `/etc/named.conf` 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` 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` flag. A non-standards-compliant variant of GSS-TSIG used by Windows 2000 can be switched on with the `-o` flag.

11.26.3 Options

-4 Use IPv4 only.

-6 Use IPv6 only.

-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.

-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`-format `key` statement, which may be generated automatically by `ddns-confgen`, or a pair of files whose names are of the format `K{name}.+157.+{random}.key` and `K{name}.+157.+{random}.private`, which can be generated by `dnssec-keygen`. 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/named/session.key`, which is automatically generated by `named` 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 zero, 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 BIND-specific resource record types whose format is understood by `nsupdate`. See also the `-T` option.

-r udpretries The number of UDP retries. The default is 3. If zero, only one update request will be made.

-t timeout The maximum time an update request can take before it is aborted. The default is 300 seconds. Zero 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 "TYPEXXXXX" 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, (<backslash> <hash> <space> <length> <space> <hexstring>).

-u udptimeout The UDP retry interval. The default is 3 seconds. If zero, 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 the version number and exit.

-y [hmac:]keyname:secret Literal TSIG authentication key. `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-md5` or if MD5 was disabled `hmac-sha256`.

NOTE: Use of the `-y` option is discouraged because the shared secret is supplied as a command line argument in clear text. This may be visible in the output from ps1 or in a history file maintained by the user's shell.

11.26.4 Input Format

`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 Dynamic DNS update request to the name server.

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 zonename` Specifies that all updates are to be made to the zone `zonename`. 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`.

ttl seconds Specify the default time to live 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 in use; the default is `hmac-md5` or if MD5 was disabled `hmac-sha256`. The `key` command overrides any key specified on the command line via `-y` or `-k`.

gsstsig Use GSS-TSIG to sign the updated. This is equivalent to specifying `-g` on the command line.

oldgsstsig Use the Windows 2000 version of GSS-TSIG to sign the updated. This is equivalent to specifying `-o` on the command line.

realm [realm_name] When using GSS-TSIG use `realm_name` rather than the default realm in `krb5.conf`. If no realm is specified the saved realm is cleared.

check-names [yes_or_no] Turn on or off check-names processing on records to be added. Check-names has no effect on prerequisites or records to be deleted. By default check-names processing is on. If check-names processing fails the record will not be added to the UPDATE message.

prereq nxdomain domain-name Requires that no resource record of any type exists with name `domain-name`.

prereq yxdomain domain-name Requires that `domain-name` exists (has as at least one resource record, of any type).

prereq nxrrset domain-name class type Requires that no resource record exists of the specified `type`, `class` and `domain-name`. If `class` is omitted, IN (internet) is assumed.

prereq yxrrset domain-name class type This requires that a resource record of the specified `type`, `class` and `domain-name` must exist. If `class` is omitted, IN (internet) is assumed.

prereq yxrrset domain-name class type data The `data` from each set of prerequisites of this form sharing a common `type`, `class`, and `domain-name` are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given `type`, `class`, and `domain-name`. The `data` are written in the standard text representation of the resource record's RDATA.

update delete domain-name ttl class type data Deletes any resource records named `domain-name`. If `type` and `data` is provided, only matching resource records will be removed. The internet class is assumed if `class` is not supplied. The `ttl` is ignored, and is only allowed for compatibility.

update add domain-name ttl class type data Adds a new resource record with the specified `ttl`, `class` and `data`.

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 version number.

help Print a list of commands.

Lines beginning with a semicolon are comments and are ignored.

11.26.5 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.)

11.26.6 Files

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

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

`K{name}.+157.+{random}.key` base-64 encoding of HMAC-MD5 key created by dnssec-keygen8.

`K{name}.+157.+{random}.private` base-64 encoding of HMAC-MD5 key created by dnssec-keygen8.

11.26.7 See Also

RFC 2136, **RFC 3007**, **RFC 2104**, **RFC 2845**, **RFC 1034**, **RFC 2535**, **RFC 2931**, *named(8)*, *ddns-confgen(8)*, *dnssec-keygen(8)*.

11.26.8 Bugs

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.

11.27 host - DNS lookup utility

11.27.1 Synopsis

host [**-aACdlnrsTUwv**] [**-c** class] [**-N** ndots] [**-p** port] [**-R** number] [**-t** type] [**-W** wait] [**-m** flag] [[**-4**] | [**-6**]] [**-v**] [**-V**] {name} [server]

11.27.2 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 name server that `host` should query instead of the server or servers listed in `/etc/resolv.conf`.

11.27.3 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 "All". The `-a` option is normally equivalent to `-v -t ANY`. It also affects the behaviour of the `-l` list zone option.

-A "Almost all". The `-A` option is equivalent to `-a` except RRSIG, NSEC, and NSEC3 records are omitted from the output.

-c class Query class: This can be used to lookup HS (Hesiod) or CH (Chaosnet) class resource records. The default class is IN (Internet).

-C Check consistency: `host` will query the SOA records for zone `name` from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone.

-d Print debugging traces. Equivalent to the `-v` verbose 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 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`.

-p port Specify the port on the server to query. The default is 53.

-r Non-recursive query: Setting this option clears the RD (recursion desired) bit in the query. This should mean that the name server receiving the query will not attempt to resolve `name`. The `-r` option enables `host` to mimic the behavior of a name server by making non-recursive queries and expecting to receive answers to those queries that can be referrals to other name servers.

-R number Number of retries for UDP queries: If `number` is negative or zero, the number of retries will default to 1. The default value is 1, or the value of the `attempts` option in `/etc/resolv.conf`, if set.

-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 Query type: The `type` argument can be any recognized query 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; -U TCP/UDP: By default, `host` uses UDP when making queries. The `-T` option makes it use a TCP connection when querying the name server. TCP will be automatically selected for queries that require it, such as zone transfer (AXFR) requests. Type ANY queries default to TCP but can be forced to UDP initially using `-U`.

-m flag Memory usage debugging: the flag can be `record`, `usage`, or `trace`. You can specify the `-m` option more than once to set multiple flags.

-v Verbose output. Equivalent to the `-d` debug option. Verbose output can also be enabled by setting the `debug` option in `/etc/resolv.conf`.

-V Print the version number and exit.

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

-W wait Timeout: Wait for up to `wait` seconds for a reply. If `wait` is less than one, the wait interval is set to one second.

By default, `host` will wait for 5 seconds for UDP responses and 10 seconds for TCP connections. These defaults can be overridden by the `timeout` option in `/etc/resolv.conf`.

See also the `-w` option.

11.27.4 IDN Support

If `host` has been built with IDN (internationalized domain name) support, it can accept and display non-ASCII domain names. `host` appropriately converts character encoding of domain name before sending a request to DNS server or displaying a reply from the server. If you'd like to turn off the IDN support for some reason, define the IDN_DISABLE environment variable. The IDN support is disabled if the variable is set when `host` runs.

11.27.5 Files

`/etc/resolv.conf`

11.27.6 See Also

dig(1), *named(8)*.

11.28 dig - DNS lookup utility

11.28.1 Synopsis

dig [@server] [**-b** address] [**-c** class] [**-f** filename] [**-k** filename] [**-m**] [**-p** port#] [**-q** name] [**-t** type] [**-v**] [**-x** addr] [**-y** [hmac:]name:key] [[**-4**] | [**-6**]] [name] [type] [class] [queryopt...]

dig [**-h**]

dig [global-queryopt...] [query...]

11.28.2 Description

`dig` is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use `dig` to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than `dig`.

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. Unlike earlier versions, the BIND 9 implementation of `dig` allows multiple lookups to be issued from the command line.

Unless it is told to query a specific name server, `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 local host.

When no command line arguments or options are given, `dig` will perform an NS query for "." (the root).

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 `-r` option disables this feature, for scripts that need predictable behaviour.

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` the specify the domain name, or use "IN." and "CH." when looking up these top level domains.

11.28.3 Simple Usage

A typical invocation of `dig` looks like:

```
dig @server name type
```

where:

`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, `dig` resolves that name before querying that name server.

If no `server` argument is provided, `dig` consults `/etc/resolv.conf`; if an address is found there, it queries the name server at that address. If either of the `-4` or `-6` options are in use, then only addresses for the corresponding transport will be tried. If no usable addresses are found, `dig` will send the query to the local host. The reply from the name server that responds is displayed.

`name` is the name of the resource record that is to be looked up.

`type` indicates what type of query is required MDASH ANY, A, MX, SIG, etc. `type` can be any valid query type. If no `type` argument is supplied, `dig` will perform a lookup for an A record.

11.28.4 Options

-4 Use IPv4 only.

-6 Use IPv6 only.

-b address[#port] Set the source IP address of the query. The `address` must be a valid address on one of the host's network interfaces, or "0.0.0.0" or "::". An optional port may be specified by appending "#<port>"

-c class Set the query class. The default `class` is IN; other classes are HS for Hesiod records or CH for Chaosnet records.

-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.

-k keyfile Sign queries using TSIG using a key read from the given file. Key files can be generated using tsig-keygen8. When using TSIG authentication with `dig`, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate `key` and `server` statements in `named.conf`.

-m Enable memory usage debugging.

-p port Send the query to a non-standard port on the server, instead of the default port 53. This option would be used to test a name server that has been configured to listen for queries on a non-standard port number.

-q name The domain name to query. This is useful to distinguish the `name` from other arguments.

-r Do not read options from `${HOME}/.digrc`. This is useful for scripts that need predictable behaviour.

-t type The resource record type to query. It can be any valid query type. If it is a resource record type supported in BIND 9, it can be given by the type mnemonic (such as "NS" or "AAAA"). The default query type is "A", unless the `-x` option is supplied to indicate a reverse lookup. 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 as "TYPEnn", where "nn" is the number of the type. If the resource record type is not supported in BIND 9, the result will be displayed as described in **RFC 3597**.

-u Print query times in microseconds instead of milliseconds.

-v Print the version number and exit.

-x addr Simplified reverse lookups, for mapping addresses to names. The `addr` is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When the `-x` is used, there is no need to provide the `name`, `class` and `type` arguments. `dig` automatically performs a lookup for a name like `94.2.0.192.in-addr.arpa` and sets the query type and class to PTR and IN respectively. IPv6 addresses are looked up using nibble format under the IP6.ARPA domain.

-y [hmac:]keyname:secret Sign queries using TSIG with the given authentication key. `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-md5` or if MD5 was disabled `hmac-sha256`.

Note: 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 output from ps1 or in a history file maintained by the user's shell.

11.28.5 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 `+cdflag`. The query options are:

`+[no]aaflag` A synonym for `+[no]aaonly`.

`+[no]aaonly` Sets the "aa" flag in the query.

`+[no]additional` Display [do not display] the additional section of a reply. The default is to display it.

`+[no]adflag` Set [do not set] the AD (authentic data) bit in the query. This 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 a OPT-OUT range. AD=0 indicate that some part of the answer was insecure or not validated. This bit is set by default.

`+[no]all` Set or clear all display flags.

`+[no]answer` Display [do not display] the answer section of a reply. The default is to display it.

`+[no]authority` Display [do not display] the authority section of a reply. The default is to display it.

`+[no]badcookie` Retry lookup with the new server cookie if a BADCOOKIE response is received.

`+[no]besteffort` Attempt to display the contents of messages which are malformed. The default is to not display malformed answers.

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

`+[no]cdflag` Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

`+[no]class` Display [do not display] the CLASS when printing the record.

`+[no]cmd` Toggles the printing of the initial comment in the output, identifying the version of `dig` and the query options that have been applied. This option always has global effect; it cannot be set globally and then overridden on a per-lookup basis. The default is to print this comment.

`+[no]comments` Toggles the display of some comment lines in the output, containing information about the packet header and OPT pseudosection, and the names of the response section. The default is to print these comments.

Other types of comments in the output are not affected by this option, but can be controlled using other command line switches. These include `+[no]cmd`, `+[no]question`, `+[no]stats`, and `+[no]rrcomments`.

`+[no]cookie=####` Send a COOKIE EDNS option, with optional value. Replaying a COOKIE from a previous response will allow the server to identify a previous client. The default is `+cookie`.

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

`+[no]crypto` 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]".

`+[no]defname` Deprecated, treated as a synonym for `+[no]search`

`+[no]dnssec` Requests DNSSEC records be sent by setting the DNSSEC OK bit (DO) 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.

+dscp=value Set the DSCP code point to be used when sending the query. Valid DSCP code points are in the range [0..63]. By default no code point is explicitly set.

+[no]edns[=#] Specify the EDNS version to query with. Valid values are 0 to 255. Setting the EDNS version will cause a EDNS query to be sent. `+noedns` clears the remembered EDNS version. EDNS is set to 0 by default.

+[no]ednsflags[=#] 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.

+[no]ednsnegotiation Enable / disable EDNS version negotiation. By default EDNS version negotiation is enabled.

+[no]ednsopt[=code[:value]] 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.

+[no]expire Send an EDNS Expire option.

+[no]fail 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.

+[no]header-only Send a query with a DNS header without a question section. The default is to add a question section. The query type and query name are ignored when this is set.

+[no]identify Show [or do not show] 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.

+[no]idnin Process [do not process] IDN domain names on input. This requires IDN SUPPORT to have been enabled at compile time.

The default is to process IDN input when standard output is a tty. The IDN processing on input is disabled when dig output is redirected to files, pipes, and other non-tty file descriptors.

+[no]idnout Convert [do not convert] puny code on output. This requires IDN SUPPORT to have been enabled at compile time.

The default is to process puny code on output when standard output is a tty. The puny code processing on output is disabled when dig output is redirected to files, pipes, and other non-tty file descriptors.

+[no]ignore Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+[no]keepalive Send [or do not send] an EDNS Keepalive option.

+[no]keepopen Keep the TCP socket open between queries and reuse it rather than creating a new TCP socket for each lookup. The default is `+nokeepopen`.

+[no]mapped Allow mapped IPv4 over IPv6 addresses to be used. The default is `+mapped`.

+[no]multiline 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=D Set the number of dots that have to appear in `name` to `D` 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.

+[no]nsid Include an EDNS name server ID request when sending a query.

+[no]nssearch 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 name server has for the zone. Addresses of servers that that did not respond are also printed.

+[no]onesoa Print only one (starting) SOA record when performing an AXFR. The default is to print both the starting and ending SOA records.

+[no]opcode=value Set [restore] the DNS message opcode to the specified value. The default value is QUERY (0).

+padding=value Pad the size of the query packet using the EDNS Padding option to blocks of `value` bytes. For example, `+padding=32` would cause a 48-byte query to be padded to 64 bytes. The default block size is 0, which disables padding. The maximum is 512. Values are ordinarily expected to be powers of two, such as 128; however, this is not mandatory. Responses to padded queries may also be padded, but only if the query uses TCP or DNS COOKIE.

+[no]qr Toggles the display of the query message as it is sent. By default, the query is not printed.

+[no]question Toggles the display of the question section of a query when an answer is returned. The default is to print the question section as a comment.

+[no]raflag Set [do not set] the RA (Recursion Available) bit in the query. The default is +noraflag. This bit should be ignored by the server for QUERY.

+[no]rdflag A synonym for `+[no]recurse`.

+[no]recurse Toggle the setting of the RD (recursion desired) bit 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.

+retry=T Sets the number of times to retry UDP queries to server to `T` instead of the default, 2. Unlike `+tries`, this does not include the initial query.

+[no]rrcomments 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.

+[no]search Use [do not use] the search list defined by the searchlist or domain directive in `resolv.conf` (if any). The search list is not used by default.

'ndots' from `resolv.conf` (default 1) 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.

+[no]short Provide a terse answer. The default is to print the answer in a verbose form. This option always has global effect; it cannot be set globally and then overridden on a per-lookup basis.

+[no]showsearch Perform [do not perform] a search showing intermediate results.

+[no]sigchase This feature is now obsolete and has been removed; use `delv` instead.

+split=W 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 multiline mode is active.

+[no]stats 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 as a comment after each lookup.

+[no]subnet=addr[/prefix-length] Send (don't send) an EDNS Client Subnet option with the specified IP address or network prefix.

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

+[no]tcflag Set [do not set] the TC (TrunCation) bit in the query. The default is +notcflag. This bit should be ignored by the server for QUERY.

+[no]tcp Use [do not use] TCP when querying name servers. The default behavior is to use UDP unless a type `any` or `ixfr=N` query is requested, in which case the default is TCP. AXFR queries always use TCP.

+timeout=T Sets the timeout for a query to `T` seconds. The default timeout is 5 seconds. An attempt to set `T` to less than 1 will result in a query timeout of 1 second being applied.

+[no]topdown This feature is related to `dig +sigchase`, which is obsolete and has been removed. Use `delv` instead.

+[no]trace Toggle tracing of the delegation path from the root name servers 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 name servers.

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

+tries=T Sets the number of times to try UDP queries to server to `T` instead of the default, 3. If `T` is less than or equal to zero, the number of tries is silently rounded up to 1.

+trusted-key=#### Formerly specified trusted keys for use with `dig +sigchase`. This feature is now obsolete and has been removed; use `delv` instead.

+[no]ttlid Display [do not display] the TTL when printing the record.

+[no]ttlunits Display [do not display] the TTL in friendly human-readable time units of "s", "m", "h", "d", and "w", representing seconds, minutes, hours, days and weeks. Implies +ttlid.

+[no]unexpected Accept [do not accept] answers from unexpected sources. By default, `dig` won't accept a reply from a source other than the one to which it sent the query.

+[no]unknownformat Print all RDATA in unknown RR type presentation format (**RFC 3597**). The default is to print RDATA for known types in the type's presentation format.

+[no]vc Use [do not use] TCP when querying name servers. This alternate syntax to `+[no]tcp` is provided for backwards compatibility. The "vc" stands for "virtual circuit".

+[no]yaml Print the responses (and, if <option>+qr</option> is in use, also the outgoing queries) in a detailed YAML format.

+[no]zflag Set [do not set] the last unassigned DNS header flag in a DNS query. This flag is off by default.

11.28.6 Multiple Queries

The BIND 9 implementation of `dig` supports specifying multiple queries on the command line (in addition to supporting the `-f` batch file 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. 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 `+[no]cmd` and `+[no]short` options) can be overridden by a query-specific set of query options. For example:

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

shows how `dig` could be used from the command line to make three lookups: an ANY query for `www.isc.org`, a reverse lookup of 127.0.0.1 and a query for the NS records of `isc.org`. A global query option of `+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 `isc.org`.

11.28.7 IDN Support

If `dig` has been built with IDN (internationalized domain name) support, it can accept and display non-ASCII domain names. `dig` appropriately converts character encoding of domain name before sending a request to DNS server or displaying a reply from the server. If you'd like to turn off the IDN support for some reason, use parameters `+noidnin` and `+noidnout` or define the IDN_DISABLE environment variable.

11.28.8 Files

`/etc/resolv.conf`

`${HOME}/.digrc`

11.28.9 See Also

delv(1), *host(1)*, *named(8)*, *dnssec-keygen(8)*, **RFC 1035**.

11.28.10 Bugs

There are probably too many query options.

11.29 nslookup - query Internet name servers interactively

11.29.1 Synopsis

nslookup [-option] [name | -] [server]

11.29.2 Description

`Nslookup` is a program to query Internet domain name servers. `Nslookup` has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

11.29.3 Arguments

Interactive mode is entered in the following cases:

a. when no arguments are given (the default name server will be used)

b. when the first argument is a hyphen (-) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen. For example, to change the default query type to host information, and the initial timeout to 10 seconds, type:

```
nslookup -query=hinfo  -timeout=10
```

The `-version` option causes `nslookup` to print the version number and immediately exits.

11.29.4 Interactive Commands

`host` [**server**] Look up information for host using the current default server or using server, if specified. If host is an Internet address and the query type is A or PTR, the name of the host is returned. If host is a name and does not have a trailing period, the search list is used to qualify the name.

To look up a host not in the current domain, append a period to the name.

`server` **domain** | **`lserver`** **domain** Change the default server to domain; `lserver` uses the initial server to look up information about domain, while `server` uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

`root` not implemented

`finger` not implemented

`ls` not implemented

`view` not implemented

`help` not implemented

`?` not implemented

`exit` Exits the program.

`set` **keyword**[=**value**] This command is used to change state information that affects the lookups. Valid keywords are:

`all` Prints the current values of the frequently used options to `set`. Information about the current default server and host is also printed.

`class=value` Change the query class to one of:

`IN` the Internet class

`CH` the Chaos class

`HS` the Hesiod class

`ANY` wildcard

The class specifies the protocol group of the information.

(Default = IN; abbreviation = cl)

nodebug Turn on or off the display of the full response packet and any intermediate response packets when searching.

(Default = nodebug; abbreviation = [no]deb)

nod2 Turn debugging mode on or off. This displays more about what nslookup is doing.

(Default = nod2)

domain=name Sets the search list to name.

nosearch If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received.

(Default = search)

port=value Change the default TCP/UDP name server port to value.

(Default = 53; abbreviation = po)

querytype=value | type=value Change the type of the information query.

(Default = A and then AAAA; abbreviations = q, ty)

Note: It is only possible to specify one query type, only the default behavior looks up both when an alternative is not specified.

norecurse Tell the name server to query other servers if it does not have the information.

(Default = recurse; abbreviation = [no]rec)

ndots=number Set the number of dots (label separators) in a domain that will disable searching. Absolute names always stop searching.

retry=number Set the number of retries to number.

timeout=number Change the initial timeout interval for waiting for a reply to number seconds.

novc Always use a virtual circuit when sending requests to the server.

(Default = novc)

nofail Try the next nameserver if a nameserver responds with SERVFAIL or a referral (nofail) or terminate query (fail) on such a response.

(Default = nofail)

11.29.5 Return Values

`nslookup` returns with an exit status of 1 if any query failed, and 0 otherwise.

11.29.6 IDN Support

If `nslookup` has been built with IDN (internationalized domain name) support, it can accept and display non-ASCII domain names. `nslookup` appropriately converts character encoding of domain name before sending a request to DNS server or displaying a reply from the server. If you'd like to turn off the IDN support for some reason, define the IDN_DISABLE environment variable. The IDN support is disabled if the variable is set when `nslookup` runs or when the standard output is not a tty.

11.29.7 Files

`/etc/resolv.conf`

11.29.8 See Also

dig(1), *host(1)*, *named(8)*.

11.30 named - Internet domain name server

11.30.1 Synopsis

named [[**-4**] | [**-6**]] [**-c** config-file] [**-d** debug-level] [**-D** string] [**-E** engine-name] [**-f**] [**-g**] [**-L** logfile] [**-M** option] [**-m** flag] [**-n** #cpus] [**-p** port] [**-s**] [**-S** #max-socks] [**-t** directory] [**-U** #listeners] [**-u** user] [**-v**] [**-V**] [**-X** lock-file] [**-x** cache-file]

11.30.2 Description

named is a Domain Name System (DNS) server, part of the BIND 9 distribution from ISC. For more information on the DNS, see **RFC 1033**, **RFC 1034**, and **RFC 1035**.

When invoked without arguments, **named** will read the default configuration file `/etc/named.conf`, read any initial data, and listen for queries.

11.30.3 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 configuration file instead of the default, `/etc/named.conf`. To ensure that reloading the configuration file continues to work after the server has changed its working directory due to 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. Debugging traces from **named** become more verbose as the debug level increases.

-D string Specifies a string that is used to identify a instance of **named** in a process listing. The contents of string are not examined.

-E engine-name When applicable, specifies the hardware to use for cryptographic operations, such as a secure key store used for signing.

When BIND is built with OpenSSL PKCS#11 support, this defaults to the string "pkcs11", which identifies an OpenSSL engine that can drive a cryptographic accelerator or hardware service module. When BIND is built with native PKCS#11 cryptography (–enable-native-pkcs11), it defaults to the path of the PKCS#11 provider library specified via "–with-pkcs11".

-f Run the server in the foreground (i.e. do not daemonize).

-g Run the server in the foreground and force all logging to `stderr`.

-L logfile Log to the file `logfile` by default instead of the system log.

-M option Sets the default memory context options. If set to external, this causes the internal memory manager to be bypassed in favor of system-provided memory allocation functions. If set to fill, blocks of memory will be filled with tag values when allocated or freed, to assist debugging of memory problems. (nofill disables this behavior, and is the default unless `named` has been compiled with developer options.)

-m flag Turn on memory usage debugging flags. Possible flags are usage, trace, record, size, and mctx. These correspond to the ISC_MEM_DEBUGXXXX flags described in `<isc/mem.h>`.

-n #cpus Create #cpus worker threads to take advantage of multiple CPUs. If not specified, `named` will try to determine the number of CPUs present and create one thread per CPU. If it is unable to determine the number of CPUs, a single worker thread will be created.

-p port Listen for queries on port port. If not specified, the default is port 53.

-s Write memory usage statistics to `stdout` on exit.

Note: This option is mainly of interest to BIND 9 developers and may be removed or changed in a future release.

-S #max-socks Allow `named` to use up to #max-socks sockets. The default value is 21000 on systems built with default configuration options, and 4096 on systems built with "configure –with-tuning=small".

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 therefore 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 fewer than the specified value because `named` reserves some file descriptors for its internal use.

-t directory Chroot to directory after processing the command line arguments, but before reading the configuration file.

Warning: 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.

-U #listeners Use #listeners worker threads to listen for incoming UDP packets on each address. If not specified, `named` will calculate a default value based on the number of detected CPUs: 1 for 1 CPU, and the number of detected CPUs minus one for machines with more than 1 CPU. This cannot be increased to a value higher than the number of CPUs. If `-n` has been set to a higher value than the number of detected CPUs, then `-U` may be increased as high as that value, but no higher. On Windows, the number of UDP listeners is hardwired to 1 and this option has no effect.

-u user Setuid to user after completing privileged operations, such as creating sockets that listen on privileged ports.

Note: On Linux, `named` uses the kernel's capability mechanism to drop all root privileges except the ability to `bind(2)` to a privileged port and set process resource limits. Unfortunately, this means that the `-u` option only works when `named` is run on kernel 2.2.18 or later, or kernel 2.3.99-pre3 or later, since previous kernels did not allow privileges to be retained after `setuid(2)`.

-v Report the version number and exit.

-V Report the version number and build options, and exit.

-X lock-file Acquire a lock on the specified file at runtime; this helps to prevent duplicate `named` instances from running simultaneously. Use of this option overrides the `lock-file` option in `named.conf`. If set to `none`, the lock file check is disabled.

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

> **Warning:** This option must not be used. It is only of interest to BIND 9 developers and may be removed or changed in a future release.

11.30.4 Signals

In routine operation, signals should not be used to control the nameserver; `rndc` should be used instead.

SIGHUP Force a reload of the server.

SIGINT, SIGTERM Shut down the server.

The result of sending any other signals to the server is undefined.

11.30.5 Configuration

The `named` configuration file is too complex to describe in detail here. A complete description is provided in the BIND 9 Administrator Reference Manual.

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

11.30.6 Files

`/etc/named.conf` The default configuration file.

`/var/run/named/named.pid` The default process-id file.

11.30.7 See Also

RFC 1033, **RFC 1034**, **RFC 1035**, *named-checkconf(8)*, *named-checkzone(8)*, *rndc(8)*, *:manpage:`named.conf(5)*, BIND 9 Administrator Reference Manual.

11.31 pkcs11-keygen - generate keys on a PKCS#11 device

11.31.1 Synopsis

`pkcs11-keygen` [**-a** algorithm] [**-b** keysize] [**-e**] [**-i** id] [**-m** module] [**-P**] [**-p** PIN] [**-q**] [**-S**] [**-s** slot] label

11.31.2 Description

`pkcs11-keygen` causes a PKCS#11 device to generate a new key pair with the given `label` (which must be unique) and with `keysize` bits of prime.

11.31.3 Arguments

-a algorithm Specify the key algorithm class: Supported classes are RSA, DSA, DH, ECC and ECX. In addition to these strings, the `algorithm` can be specified as a DNSSEC signing algorithm that will be used with this key; for example, NSEC3RSASHA1 maps to RSA, ECDSAP256SHA256 maps to ECC, and ED25519 to ECX. The default class is "RSA".

-b keysize Create the key pair with `keysize` bits of prime. For ECC keys, the only valid values are 256 and 384, and the default is 256. For ECX keys, the only valid values are 256 and 456, and the default is 256.

-e For RSA keys only, use a large exponent.

-i id Create key objects with id. The id is either an unsigned short 2 byte or an unsigned long 4 byte number.

-m module Specify the PKCS#11 provider module. This must be the full path to a shared library object implementing the PKCS#11 API for the device.

-P Set the new private key to be non-sensitive and extractable. The allows the private key data to be read from the PKCS#11 device. The default is for private keys to be sensitive and non-extractable.

-p PIN Specify the PIN for the device. If no PIN is provided on the command line, `pkcs11-keygen` will prompt for it.

-q Quiet mode: suppress unnecessary output.

-S For Diffie-Hellman (DH) keys only, use a special prime of 768, 1024 or 1536 bit size and base (aka generator) 2. If not specified, bit size will default to 1024.

-s slot Open the session with the given PKCS#11 slot. The default is slot 0.

11.31.4 See Also

pkcs11-destroy(8), *pkcs11-list(8)*, *pkcs11-tokens(8)*, *dnssec-keyfromlabel(8)*

11.32 pkcs11-tokens - list PKCS#11 available tokens

11.32.1 Synopsis

pkcs11-tokens [**-m** module] [**-v**]

11.32.2 Description

`pkcs11-tokens` lists the PKCS#11 available tokens with defaults from the slot/token scan performed at application initialization.

11.32.3 Arguments

-m module Specify the PKCS#11 provider module. This must be the full path to a shared library object implementing the PKCS#11 API for the device.

-v Make the PKCS#11 libisc initialization verbose.

11.32.4 See Also

pkcs11-destroy(8), *pkcs11-keygen(8)*, *pkcs11-list(8)*

11.33 pkcs11-list - list PKCS#11 objects

pkcs11-list [**-P**] [**-m** module] [**-s** slot] [**-i** ID] [**-l** label] [**-p** PIN]

11.33.1 Description

`pkcs11-list` lists the PKCS#11 objects with `ID` or `label` or by default all objects. The object class, label, and ID are displayed for all keys. For private or secret keys, the extractability attribute is also displayed, as either `true`, `false`, or `never`.

11.33.2 Arguments

-P List only the public objects. (Note that on some PKCS#11 devices, all objects are private.)

-m module Specify the PKCS#11 provider module. This must be the full path to a shared library object implementing the PKCS#11 API for the device.

-s slot Open the session with the given PKCS#11 slot. The default is slot 0.

-i ID List only key objects with the given object ID.

-l label List only key objects with the given label.

-p PIN Specify the PIN for the device. If no PIN is provided on the command line, `pkcs11-list` will prompt for it.

11.33.3 See Also

pkcs11-destroy(8), *pkcs11-keygen(8)*, *pkcs11-tokens(8)* pkcs11-destroy - destroy PKCS#11 objects

11.33.4 Synopsis

pkcs11-destroy [**-m** module] [**-s** slot] [**-i** ID] [**-l** label] [**-p** PIN] [**-w** seconds]

11.33.5 Description

`pkcs11-destroy` destroys keys stored in a PKCS#11 device, identified by their `ID` or `label`.

Matching keys are displayed before being destroyed. By default, there is a five second delay to allow the user to interrupt the process before the destruction takes place.

11.33.6 Arguments

-m module Specify the PKCS#11 provider module. This must be the full path to a shared library object implementing the PKCS#11 API for the device.

-s slot Open the session with the given PKCS#11 slot. The default is slot 0.

-i ID Destroy keys with the given object ID.

-l label Destroy keys with the given label.

-p PIN Specify the PIN for the device. If no PIN is provided on the command line, `pkcs11-destroy` will prompt for it.

-w seconds Specify how long to pause before carrying out key destruction. The default is five seconds. If set to `0`, destruction will be immediate.

11.33.7 See Also

pkcs11-keygen(8), *pkcs11-list(8)*, *pkcs11-tokens(8)*

11.34 named-checkconf - named configuration file syntax checking tool

11.34.1 Synopsis

`named-checkconf` [**-chjlvz**] [**-p** [**-x**]] [**-t** directory] {filename}

11.34.2 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/named.conf` is read by default.

Note: files that `named` reads in separate parser contexts, such as `rndc.key` and `bind.keys`, 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.

11.34.3 Options

-h Print the usage summary and exit.

-j When loading a zonefile read the journal if it exists.

-l List all the configured zones. Each line of output contains the zone name, class (e.g. IN), view, and type (e.g. master or slave).

-c Check "core" configuration only. This suppresses the loading of plugin modules, and causes all parameters to `plugin` statements to be ignored.

-i Ignore warnings on deprecated options.

-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 to `directory` so that include directives in the configuration file are processed as if run by a similarly chrooted `named`.

-v Print the version of the `named-checkconf` program and exit.

-x When printing the configuration files in canonical form, obscure shared secrets by replacing them with strings of question marks ('?'). This allows the contents of `named.conf` and related files to be shared MDASH for example, when submitting bug reports MDASH without compromising private data. This option cannot be used without `-p`.

-z Perform a test load of all master zones found in `named.conf`.

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

11.34.4 Return Values

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

11.34.5 See Also

named(8), *named-checkzone(8)*, BIND 9 Administrator Reference Manual.

11.35 named-checkzone, named-compilezone - zone file validity checking or converting tool

11.35.1 Synopsis

`named-checkzone` [**-d**] [**-h**] [**-j**] [**-q**] [**-v**] [**-c** class] [**-f** format] [**-F** format] [**-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-compilezone` [**-d**] [**-j**] [**-q**] [**-v**] [**-c** class] [**-C** mode] [**-f** format] [**-F** format] [**-J** filename] [**-i** mode] [**-k** mode] [**-m** mode] [**-n** mode] [**-l** ttl] [**-L** serial] [**-r** mode] [**-s** style] [**-t** directory] [**-T** mode] [**-w** directory] [**-D**] [**-W** mode] {**-o** filename} {zonename} {filename}

11.35.2 Description

`named-checkzone` checks the syntax and integrity of a zone 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 into a name server.

`named-compilezone` is similar to `named-checkzone`, but it always dumps the zone contents to a specified file in a specified format. Additionally, it applies stricter check levels by default, since the dump output will be used as an actual zone file loaded by `named`. When manually specified otherwise, the check levels must at least be as strict as those specified in the `named` configuration file.

11.35.3 Options

-d Enable debugging.

-h Print the usage summary and exit.

-q Quiet mode - exit code only.

-v Print the version of the `named-checkzone` program and exit.

-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. (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 modes 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.

-f format Specify the format of the zone file. Possible formats are `"text"` (default), `"raw"`, and `"map"`.

-F format Specify the format of the output file specified. For `named-checkzone`, this does not cause any effects unless it dumps the zone contents.

Possible formats are `"text"` (default), which is the standard textual representation of the zone, and `"map"`, `"raw"`, and `"raw=N"`, which store the zone in a binary format for rapid loading by `named`. `"raw=N"` specifies the format version of the raw zone file: if N is 0, the raw file can be read by any version of `named`; if N is 1, the file can be read by release 9.9.0 or higher; the default is 1.

-k mode Perform `"check-names"` checks with the specified failure mode. Possible modes are `"fail"` (default for `named-compilezone`), `"warn"` (default for `named-checkzone`) 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`.

-L serial When compiling a zone to "raw" or "map" format, 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"` (default for `named-compilezone`), `"warn"` (default for `named-checkzone`) and `"ignore"`.

-o filename Write zone output to `filename`. If `filename` is `-` then write to standard out. This is mandatory for `named-compilezone`.

-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 styles 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. It also does not have any meaning if the output format is not text.

-S mode Check if a SRV record refers to a CNAME. Possible modes are `"fail"`, `"warn"` (default) and `"ignore"`.

-t directory Chroot to `directory` so that include directives in the configuration file are processed as if run by a similarly chrooted `named`.

-T mode Check if Sender Policy Framework (SPF) records exist and issues a warning if an SPF-formatted TXT record is not also present. Possible modes are `"warn"` (default), `"ignore"`.

-w directory chdir to `directory` so that relative filenames in master file $INCLUDE directives work. This is similar to the directory clause in `named.conf`.

-D Dump zone file in canonical format. This is always enabled for `named-compilezone`.

-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.

11.35.4 Return Values

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

11.35.5 See Also

named(8), *named-checkconf(8)*, **RFC 1035**, BIND 9 Administrator Reference Manual.

Index

S

Y