SOFTWARE USER MANUAL (SUM): TRAINING, PROCEDURAL, AND DEVELOPMENT DOCUMENTATION

DNSSEC-Tools Software User Manual

Manual Pages

14 September 2007

Version 1.3

DNSSEC-Tools
Is your domain secure?

Contents

1 About This Document **4**

1.1 **Conventions** 5

1.2 **Comments** 5

2 DNSSEC-Tools Commands **6**

2.1 **DNSSEC-Tools Maintenance Commands** 7

2.1.1 dtinitconf 8

2.1.2 dtdefs 12

2.1.3 dtconf 13

2.1.4 dtck 14

2.1.5 dtconfchk 16

2.1.6 trustman 20

2.1.7 keyarch 23

2.1.8 timetrans 25

2.2 **DNS Zone File Commands** 27

2.2.1 dnspktflow 28

2.2.2 donuts 31

2.2.3 donutsd 34

2.2.4 getaddr 37

2.2.5 getdnskeys 38

2.2.6 gethost 40

2.2.7 libval_check_conf 41

2.2.8 mapper 42

2.2.9 tachk 45

2.2.10 validate 47

2.3 **Zone-Signing Commands** 50

2.3.1 zonesigner 51

2.3.2 genkrf . . . 62
2.3.3 krfcheck . . . 65
2.3.4 lskrf . . . 68
2.3.5 expchk . . . 77
2.3.6 fixkrf . . . 78
2.3.7 cleanarch . . . 79
2.3.8 cleankrf . . . 82
2.3.9 signset-editor . . . 83

2.4 **Zone-Rollover Commands** . . . 87
2.4.1 rollerd . . . 88
2.4.2 rollinit . . . 93
2.4.3 rollchk . . . 99
2.4.4 rollset . . . 101
2.4.5 rollrec-editor . . . 103
2.4.6 lsroll . . . 110
2.4.7 rollctl . . . 114
2.4.8 rolllog . . . 117
2.4.9 blinkenlights . . . 118

3 DNSSEC Library Routines **127**
3.0.10 libsres . . . 128
3.0.11 libval . . . 133
3.0.12 val_getaddrinfo() . . . 143
3.0.13 val_gethostbyname() . . . 146
3.0.14 val_query() . . . 150

4 Supporting Modules **153**
4.0.15 BootStrap.pm . . . 154
4.0.16 QWPrimitives.pm . . . 155
4.0.17 conf.pm . . . 156
4.0.18 defaults.pm . . . 159
4.0.19 dnssectools.pm . . . 164

4.0.20 keyrec.pm 166
4.0.21 rolllog.pm 174
4.0.22 rollmgr.pm 177
4.0.23 rollrec.pm 181
4.0.24 timetrans.pm 187
4.0.25 tooloptions.pm 189

5 Data Files **193**

5.0.26 dnssec-tools.conf 194
5.0.27 dnsval.conf 199
5.0.28 Keyrec Files 203
5.0.29 Rollrec Files 208
5.0.30 Rule.pm for **donuts** 211
5.0.31 blinkenlights.conf 216

1 About This Document

The goal of the DNSSEC-Tools project is to create a set of tools, libraries, patches, applications, wrappers, extensions, and plugins that will help ease the deployment and maintenance of DNSSEC-related technologies. This document contains manual pages for the commands, libraries, Perl modules, and data files that are part of the DNSSEC-Tools distribution. The document organization is described below.

Section 1 describes the DNSSEC-Tools Software User Manual.

Section 2 describes the DNSSEC-Tools commands. These commands include programs to analyze and manipulate zone files, maintain the DNSSEC-Tools environment, and to assist with zone signing and key rollover. The commands are divided into functional groups: DNSSEC-Tools maintenance commands (Section 2.1), DNS zone file commands (Section 2.2), zone-signing commands (Section 2.3), and zone-rollover commands (Section 2.4).

Section 3 describes two libraries developed for DNSSEC-Tools. The *libsres* library provides secure address resolution for applications. The *libval* library provides DNSSEC Resource Record validation.

Section 4 describes a number of Perl modules that were written to support the DNSSEC-Tools commands. These modules may be used in development of additional commands to work with the existing DNSSEC-Tools.

Section 5 describes data files used by the DNSSEC-Tools commands, libraries, and modules.

For more information about this project and the tools that are being developed and provided, please see one of the project web pages at:

http://www.dnssec-tools.org
http://dnssec-tools.sourceforge.net

1.1 Conventions

The following typographical conventions are used in this document.

command	Command names
constant	Code constants
call()	System and function calls
library	Library names
module	Perl Modules
path	File and path names
URL	Web URLs
variable	Variables
execution	Simple command executions

Longer sets of command sequences are given in this format:

```
# cd /tmp
# ls
# rm -fr *
```

In most cases, output will not be displayed for given command sequences.

1.2 Comments

Please send any comments and corrections to developers@dnssec-tools.org.

2 DNSSEC-Tools Commands

A number of commands have been developed to assist in maintaining DNSSEC-secured domains. These commands check zone files for errors, assist in key generation and zone signing, perform key rollover, and provide graphic information about zones.

The DNSSEC-Tools commands are divided in four functional groups:

- DNSSEC-Tools Maintenance Commands
- DNS Zone-File Commands
- Zone-Signing Commands
- Key-Rollover Commands

The commands in each functional group are described in the following subsections.

2.1 DNSSEC-Tools Maintenance Commands

The commands in this group primarily deal with DNSSEC-Tools configuration file. There is also a tool for time conversions, which may assist in determining values for some fields for a zone. The maintenance commands are:

dtinitconf	create a new DNSSEC-Tools configuration file
dtdefs	print the DNSSEC-Tools configuration values
dtconf	displays the contents of a DNSSEC-Tools configuration file
dtck	checks DNSSEC-Tools data files for sanity
dtconfchk	checks a DNSSEC-Tools configuration file for errors
trustman	daemon to manage trust anchor keys
keyarch	daemon to archive old KSK and ZSK keys
timetrans	converts time units

2.1.1 dtinitconf

NAME

dtinitconf - Creates a DNSSEC-Tools configuration file

SYNOPSIS

```
  dtinitconf [options]
```

DESCRIPTION

The **dtinitconf** program initializes the DNSSEC-Tools configuration file. By default, the actual configuration file will be created, though the created file can be specified by the user. Existing files, whether the default or one specified by the user, will not be overwritten unless specifically directed by the user.

Each configuration field can be individually specified on the command line. The user will also be prompted for the fields, with default values taken from the DNSSEC-Tools **defaults.pm** module. If the *-noprompt* option is given, then a default configuration file (modulo command-line arguments) will be created.

Configuration entries are created for several BIND programs. Several locations on the system are searched to find the locations of these programs. First, the directories in the path environment variable are checked; the names of any directories that contain the BIND programs are saved. Next, several common locations for BIND programs are checked; again, the names of directories that contain the BIND programs are saved. After collecting these directories, the user is presented with this list and may choose to use whichever set is desired. If no directories are found that contain the BIND programs, the user is prompted for the proper location.

If the configuration file's parent directory does not exist, then an attempt is made to create the directory. The new directory's ownership will be set to *root* for the owner and *dnssec* for the group, assuming the *dnssec* group exists.

OPTIONS

dtinitconf takes options that control the contents of the newly generated DNSSEC-Tools configuration file. Each configuration file entry has a corresponding command-line option. The options, described below, are ordered in logical groups.

Key-related Options

These options deal with different aspects of creating and managing encryption keys.

-algorithm algorithm

Selects the cryptographic algorithm. The value of algorithm must be one that is recognized by **dnssec-keygen**.

-ksklength keylen

The default KSK key length to be passed to **dnssec-keygen**.

-ksklife lifespan

The default length of time between KSK roll-overs. This is measured in seconds.

This value is **only** used for key roll-over. Keys do not have a life-time in any other sense.

-zskcount ZSK-count

The default number of ZSK keys that will be created for a zone.

-zsklength keylen

The default ZSK key length to be passed to **dnssec-keygen**.

-zsklife lifespan

The default length of time between ZSK roll-overs. This is measured in seconds.

This value is **only** used for key roll-over. Keys do not have a life-time in any other sense.

-random randomdev

The random device generator to be passed to **dnssec-keygen**.

Zone-related Options

These options deal with different aspects of zone signing.

-endtime endtime

The zone default expiration time to be passed to **dnssec-signzone**.

DNSSEC-Tools Options

These options deal specifically with functionality provided by DNSSEC-Tools.

-admin email-address

admin is the email address of the DNSSEC-Tools administrator. This is the default address used by the *dt_adminmail()* routine.

-archivedir directory

directory is the archived-key directory. Old encryption keys are moved to this directory, but only if they are to be saved and not deleted.

-binddir directory

directory is the directory holding the BIND programs.

-entropy_msg

A flag indicating that **zonesigner** should display a message about entropy generation. This is primarily dependent on the implementation of a system's random number generation.

-noentropy_msg

A flag indicating that **zonesigner** should not display a message about entropy generation. This is primarily dependent on the implementation of a system's random number generation.

-roll-logfile logfile

logfile is the logfile for the **rollerd** daemon.

-roll-loglevel loglevel

loglevel is the logging level for the **rollerd** daemon.

-roll-sleep sleep-time

sleep-time is the sleep-time for the **rollerd** daemon.

-savekeys

A flag indicating that old keys should be moved to the archive directory.

-nosavekeys

A flag indicating that old keys should not be moved to the archive directory but will instead be left in place.

-usegui

A flag indicating that the GUI for specifying command options may be used.

-nousegui

A flag indicating that the GUI for specifying command options should not be used.

dtinitconf Options

These options deal specifically with **dtinitconf**.

-outfile conffile

The configuration file will be written to **conffile**. If this is not given, then the default configuration file (as returned by **Net::DNS::SEC::Tools::conf::getconffile()**) will be used.

If **conffile** is given as **-**, then the new configuration file will be written to the standard output.

conffile must be writable.

-overwrite

If *-overwrite* is specified, existing output files may be overwritten. Without *-overwrite*, if the output file is found to exist then **dtinitconf** will give an error message and exit.

-noprompt

If *-noprompt* is specified, the user will not be prompted for any input. The configuration file will be created from command-line options and DNSSEC-Tools defaults. Guesses will be made for the BIND paths, based on the PATH environment variable.

WARNING: After using the *-noprompt* option, the configuration file **must** be checked to ensure that the defaults are appropriate and acceptable for the installation.

-edit

If *-edit* is specified, the output file will be edited after it has been created. The EDITOR environment variable is consulted for the editor to use. If the EDITOR environment variable isn't defined, then the **vi** editor will be used.

-verbose

Provide verbose output.

-help

Display a usage message and exit.

SEE ALSO

dnssec-keygen(8), dnssec-signzone(8), named-checkzone(8), rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3), Net::DNS::SEC::Tools::dnssectools.pm(3), Net::DNS::SEC::Tools::tooloptions.pm(3), QWizard.pm(3)

2.1.2 dtdefs

NAME

dtdefs - Displays defaults defined for DNSSEC-Tools

SYNOPSIS

```
  dtdefs
```

DESCRIPTION

The **dtdefs** program displays defaults defined for DNSSEC-Tools.

SEE ALSO

Net::DNS::SEC::Tools::defaults.pm(3)

2.1.3 dtconf

NAME

dtconf - Display the contents of a DNSSEC-Tools configuration file

SYNOPSIS

```
  dtconf [options] [config_file]
```

DESCRIPTION

dtconf displays the key/value pairs of a DNSSEC-Tools configuration file. If a configuration file isn't specified, the system configuration file will be displayed.

Without the **-key** option, **dtconf** displays all the key/value pairs in the configuration file. Comments are never displayed. If the **-key** option is given, then only that key/value pair is displayed. If the key isn't defined, then the value will be "(undefined)."

OPTIONS

-key

The value of the specified key will be printed. If the key is not defined, then the value will be given as "(undefined)".

-Version

Version information will be printed.

-help

Display a usage message.

SEE ALSO

dtinitconf(8), dtconfchk(8)

Net::DNS::SEC::Tools::conf.pm(3)

2.1.4 dtck

NAME

dtck - Check the DNSSEC-Tools data files for sanity

SYNOPSIS

```
  dtck [options] [dtck_config_file]
```

DESCRIPTION

dtck checks DNSSEC-Tools data files to determine if the entries are valid. **dtck** checks the validity of DNSSEC-Tools configuration files, *rollrec* files, and *keyrec* files. It does not perform the file checking itself, but runs checking programs specific to each type of data file.

A **dtck** configuration file is consulted to determine the files to check. This file lists the DNSSEC-Tools data files and their types. If a **dtck** configuration file is not given on the command line, **dtck** will only check the DNSSEC-Tools configuration file. This is equivalent to running **dtconfchk** directly.

DTCK CONFIGURATION FILE

A **dtck** configuration file contains a list of the files to be checked by **dtck**. Except for comments, each line has the following format:

```
    keyword file directory
```

keyword is one of "config", "rollrec", or "keyrec". *file* is the pathname of the file to be checked. *directory* is the name of the directory that holds *file* and is optional.

The **dtck** configuration file contains the following types of records:

config

These lines define the DNSSEC-Tools configuration files that will be checked. The **dtconfchk** program will be used to verify these files.

rollrec

These lines define the *rollrec* files that will be checked. The **rollchk** program will be used to verify these files.

keyrec

These lines define the *keyrec* files that will be checked. The **krfcheck** program will be used to verify these files.

comments

Any lines starting with an octothorpe (#) are comment lines and are ignored.

OPTIONS

dtck takes two types of options. Options of the first type are handled directly by **dtck**, controlling its output and processing. Options of the second type are passed to the file-checking programs and are not further handled by **dtck**.

Options Handled by dtck

-defcon

This option directs **dtck** to add the default DNSSEC-Tools configuration file to the list of configuration files to be checked.

-list

The names of the files will be listed as they are checked.

-pretty

Clarifying output is added to the output from **dtck** and the file-checking programs.

-help

Display a usage message.

Options Not Handled by dtck

-count

The file-checking programs will display a final error count.

-quiet

No output will be given by the file-checking program.

-verbose

Verbose output will be given by the file-checking program.

SEE ALSO

dtconfchk(8), krfcheck(8), rollchk(8)

dnssec-tools.conf(5), keyrec(5), rollrec(5)

2.1.5 dtconfchk

NAME

dtconfchk - Check a DNSSEC-Tools configuration file for sanity

SYNOPSIS

```
  dtconfchk [options] [config_file]
```

DESCRIPTION

dtconfchk checks a DNSSEC-Tools configuration file to determine if the entries are valid. If a configuration file isn't specified, the system configuration file will be verified.

Without any display options, **dtconfchk** displays error messages for problems found, followed by a summary line. Display options will increase or decrease the amount of detail about the configuration file's sanity. In all cases, the exit code is the count of errors found in the file.

The tests are divided into five groups: key-related checks, zone-related checks, path checks, rollover checks, and miscellaneous checks. The checks in each of these self-explanatory groups are described below.

The *default_keyrec* configuration entry is not checked. This entry specifies the default *keyrec* file name and isn't necessarily expected to exist in any particular place.

Key-related Checks

The following key-related checks are performed:

- *algorithm*

  Ensure the *algorithm* field is valid. The acceptable values may be found in the **dnssec-keygen** man page.

- *ksklength*

  Ensure the *ksklength* field is valid. The acceptable values may be found in the **dnssec-keygen** man page.

- *ksklife*

  Ensure the *ksklife* field is valid. The acceptable values may be found in the **defaults.pm** man page.

- *zskcount*

  Ensure the *zskcount* field is valid. The ZSK count must be positive.

- *zsklength*

  Ensure the *zsklength* field is valid. The acceptable values may be found in the **dnssec-keygen** man page.

- *zsklife*

  Ensure the *zsklife* field is valid. The acceptable values may be found in the **defaults.pm** man page.

- *random*

  Ensure the *random* field is valid. This file must be a character device file.

Zone-related Checks

The following zone-related checks are performed:

endtime

  Ensure the *endtime* field is valid. This value is assumed to be in the "+NNNNNN" format. There is a lower limit of two hours. (This is an artificial limit under which it *may* not make sense to have an end-time.)

Path Checks

The following path checks are performed:

- *keygen*

  Ensure the *keygen* field is valid. If the filename starts with a '/', the file must be a regular executable file.

- *viewimage*

  Ensure the *viewimage* field is valid. If the filename starts with a '/', the file must be a regular executable file.

- *zonecheck*

  Ensure the *zonecheck* field is valid. If the filename starts with a '/', the file must be a regular executable file.

- *zonesign*

  Ensure the *zonesign* field is valid. If the filename starts with a '/', the file must be a regular executable file.

Rollover Daemon Checks

The following checks are performed for **rollerd** values:

- *roll_logfile*

  Ensure that the log file for the **rollerd** is valid. If the file exists, it must be a regular file.

Textual Level	Numeric Level	Meaning
tmi	1	Overly verbose informational messages.
expire	3	A verbose countdown of zone expiration is given.
info	4	Informational messages.
phase	6	Current state of zone.
err	8	Errors messages.
fatal	9	Fatal errors.

Table 1: Logging Levels

- *roll_loglevel*

  Ensure that the logging level for the **rollerd** is reasonable. The log level must be one of the following text or numeric values:

  Specifying a particular log level will causes messages of a higher numeric value to also be displayed.

- *roll_sleeptime*

  Ensure that the **rollerd**'s sleep-time is reasonable. **rollerd**'s sleep-time must be at least one minute.

Miscellaneous Checks

The following miscellaneous checks are performed:

- *admin-email*

  Ensure that the *admin-email* field is defined and has a value. **dtconfchk** does not try to validate the email address itself.

- *archivedir*

  Ensure that the *archivedir* directory is actually a directory. This check is only performed if the *savekeys* flag is set on.

- *entropy_msg*

  Ensure that the *entropy_msg* flag is either 0 or 1.

- *savekeys*

  Ensure that the *savekeys* flag is either 0 or 1. If this flag is set to 1, then the *archivedir* field will also be checked.

- *usegui*

  Ensure that the *usegui* flag is either 0 or 1.

OPTIONS

-expert

This option will bypass the following checks:

- KSK has a longer lifespan than the configuration file's default minimum lifespan
- KSK has a shorter lifespan than the configuration file's default maximum lifespan
- ZSKs have a longer lifespan than the configuration file's default minimum lifespan
- ZSKs have a shorter lifespan than the configuration file's default maximum lifespan

-quiet

No output will be given. The number of errors will be used as the exit code.

-summary

A final summary of success or failure will be printed. The number of errors will be used as the exit code.

-verbose

Success or failure status of each check will be given. A + or **-** prefix will be given for each valid and invalid entry. The number of errors will be used as the exit code.

-help

Display a usage message.

SEE ALSO

dtdefs(8), dtinitconf(8), rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3)

dnssec-tools.conf(5)

2.1.6 trustman

NAME

trustman - Manage keys used as trust anchors

SYNOPSIS

trustman [options]

DESCRIPTION

trustman manages keys used by DNSSEC as trust anchors. It may be used as a daemon for ongoing key verification or manually for initialization and one-time key verification.

By default, **trustman** runs as a daemon to ensure that keys stored locally in configuration files still match the same keys fetched from the zone where they are defined. (**named.conf** and **dnsval.conf** are the usual configuration files.) These checks can be run once manually (*-S*) and in the foreground (*-f*).

For each key mismatch check, if key mismatches are detected then **trustman** performs the following operations:

- sets an add hold-down timer for new keys;
- sets a remove hold-down timer for missing keys;
- removes revoked keys from the configuration file.

On subsequent runs, the timers are checked. If the timers have expired, keys are added or removed from the configuration file, as appropriate.

CONFIGURATION

trustman can also set up configuration data in the DNSSEC-Tools configuration file for later use by the daemon. This makes fewer command line arguments necessary on subsequent executions. (The configuration file is in **dnssec-tools.conf**.)

Configuration data is stored in **dnssec-tools.conf**. The current version requires you to edit **dnssec-tools.conf** by hand and supply values for the contact person's email address (*tacontact*) and the SMTP server (*tasmtpserver*). If necessary, edit the location of **named.conf** and **dnsval.conf** in that file.

OPTIONS

trustman takes a number of options, each of which is described in this section. Each option name may be shortened to the minimum number of unique characters, but some options also have an alias (as noted.) The single-letter form of each option is denoted in parentheses, e.g., *–anchor_data_file datafile (-a)*.

–anchor_data_file datafile (-a)

A persistent data file for storing new keys waiting to be added.

–config (-c)

Create a configure file for **trustman** from the command line options given.

–dnsval_conf_file conffile (-k)

A **dnsval.conf** file to read.

–zone zone (-z)

The zone to check. Specifying this option supersedes the default configuration file.

–foreground (-f)

Run in the foreground.

–hold_time seconds (-w)

The value of the hold-down timer.

–mail_contact_addr email-address (-m)

Mail address for the contact person to whom reports should be sent.

–named_conf_file conffile (-n)

A **named.conf** file to read.

–no_error (-N)

Send report when there are no errors.

–outfile output-file (-o)

Output file for configuration.

–print (-p)

Log/print messages to stdout.

–resolv_conf_file conffile (-r)

A **resolv.conf** file to read. **/dev/null** can be specified to force *libval* to recursively answer the query rather than asking other name servers).

–smtp_server smtpservername (-s)

SMTP server that **trustman** should use to send reports by mail.

–single_run (-S)

Run only once.

–syslog (-L)

Log messages to **syslog**.

–sleeptime seconds (-t)

The number of seconds to sleep between checks. Default is 3600 (one hour.)

–test_revoke

Use this option to test the REVOKE bit. No known implementation of the REVOKE bit exists to date.

–help (-h)

Display a help message.

–verbose (-v)

Verbose output.

–version (-V)

Display version information.

SEE ALSO

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3),

dnssec-tools.conf(5)

2.1.7 keyarch

NAME

keyarch - DNSSEC-Tools daemon to archive old KSK and ZSK keys

SYNOPSIS

```
  keyarch [options] <keyrec_file | rollrec_file>
```

DESCRIPTION

The **keyarch** program archives old KSK and ZSK keys. Keys are considered old if they are obsolete and are marked as either *kskobs* or *zskobs*. Archived keys are prefixed with the seconds-since-epoch as a means of distinguishing a zone's keys that have the same five digit number.

If the required file argument is a *keyrec* file, then expired keys listed in that file are archived. If the file argument is a *rollrec* file, the *keyrec* files of the zones in that file are checked for expired keys.

If the *-zone* option is given, then only obsolete keys belonging to the specified zone will be archived.

The archive directory is either zone-specific (listed in the zone's *keyrec* record in the zone's *keyrec* file) or the default archive directory given in the DNSSEC-Tools configuration file.

The count of archived keys is given as the program's exit code. Error exit codes are negative.

OPTIONS

The following options are recognized:

-zone zone_file

Name of the zone whose KSKs will be archived. If this is not given, then all the zones defined in the *rollrec* file will be checked.

-kskonly

Only archive KSK keys.

-zskonly

Only archive ZSK keys.

-quiet

No output will be given.

-verbose

Verbose output will be given.

-help

Display a usage message.

-Version
Display the program versions.

EXIT VALUES

On success, **keyarch**'s exit code is the number of keys archived.

keyarch has a 0 exit code if the help message is given.

keyarch has a negative exit code if an error is encountered.

SEE ALSO

rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::dnssectools.pm(3),
Net::DNS::SEC::Tools::defaults.pm(3), Net::DNS::SEC::Tools::keyrec.pm(3),
Net::DNS::SEC::Tools::rollrec.pm(3)

keyrec(5), rollrec(5)

2.1.8 timetrans

NAME

timetrans - Converts time into time

SYNOPSIS

```
  timetrans [units-options] [-count]
```

DESCRIPTION

timetrans converts time from one type of unit to another. If any of the units options are specified, then **timetrans** will convert those time units into the number of seconds to which they add up. If given the count option, **timetrans** will convert that number of seconds into the appropriate number of weeks, days, hours, minutes, and seconds. The converted result is printed out. Units options cannot be specified in the same execution as the count option, and vice versa.

timetrans is intended for use with DNSSEC-Tools, for calculating a zone's expiration time.

OPTIONS

Units Options

The converted value of each unit is totaled and a single result printed.

-seconds seconds

Count of seconds to convert to seconds.

-minutes minutes

Count of minutes to convert to seconds.

-hours hours

Count of hours to convert to seconds.

-days days

Count of days to convert to seconds.

-weeks weeks

Count of weeks to convert to seconds.

Count Option

The specified seconds count is converted to the appropriate number of weeks, days, hours, minutes, and seconds.

-count seconds

Count of seconds to convert to the appropriate set of units.

EXAMPLES

Example 1: Converting 5 days into seconds

```
    $(42)> timetrans -days 5
    432000
```

Example 2: Converting 2 weeks into seconds

```
    $(43)> timetrans -w 2
    1209600
```

Example 3: Converting 8 days and 8 hours into seconds

```
    $(44)> timetrans -d 8 -hours 8
    720000
```

Example 4: Converting 1 week, 1 day, and 8 hours into seconds

```
    $(46)> timetrans -w 1 -days 1 -h 8
    720000
```

Example 5: Converting 14 weeks, 4 days, 21 hours, 8 minutes, and 8 seconds into seconds

```
    $(47)> timetrans -w 14 -d 4 -h 21 -m 8 -s 8
    8888888
```

Example 6: Converting 720000 seconds into time units

```
    $(48)> timetrans -c 720000
    1 week, 1 day, 8 hours
```

Example 7: Converting 1814421 seconds into time units

```
    $(49)> timetrans -c 1814421
    3 weeks, 21 seconds
```

Example 8: Converting 8888888 seconds into time units

```
    $(50)> timetrans -c 8888888
    14 weeks, 4 days, 21 hours, 8 minutes, 8 seconds
```

SEE ALSO

zonesigner(8)

Net::DNS::SEC::Tools::timetrans.pm(3)

2.2 DNS Zone File Commands

The DNS zone file commands provide analytical and visualization tools for DNS zone files. These commands are:

dnspktflow	analyzes and draw DNS flow diagrams
donuts	analyzes DNS zone files for errors
donutsd	daemon to periodically run **donuts**
getaddr	test program for *val_getaddrinfo()*
getdnskeys	manage lists of DNSKEYs from DNS zones
gethost	test program for *val_gethostbyname()*
libval_check_conf	verifies validator configuration file
mapper	creates maps of DNS zone data
tachk	verifies trust anchors in a **named.conf** file
validate	query the Domain Name System

2.2.1 dnspktflow

NAME

dnspktflow - Analyze and draw DNS flow diagrams from a **tcpdump** file

SYNOPSIS

```
    dnspktflow -o output.png file.tcpdump

    dnspktflow -o output.png -x -a -t -q file.tcpdump
```

DESCRIPTION

The **dnspktflow** application takes a **tcpdump** network traffic dump file, passes it through the **tshark** application and then displays the resulting DNS packet flows in a "flow-diagram" image. **dnspktflow** can output a single image or a series of images which can then be shown in sequence as an animation.

dnspktflow was written as a debugging utility to help trace DNS queries and responses, especially as they apply to DNSSEC-enabled lookups.

REQUIREMENTS

This application requires the following Perl modules and software components to work:

```
     graphviz                  http://www.graphviz.org/
     GraphViz                  Perl module
     tshark                    http://www.wireshark.org/
```

The following is required for outputting screen presentations:

```
    MagicPoint                 http://member.wide.ad.jp/wg/mgp/
```

If the following modules are installed, a GUI interface will be enabled for communication with **dnspktflow**:

```
     QWizard                   Perl module
     Getopt::GUI::Long         Perl module
```

OPTIONS

dnspktflow takes a wide variety of command-line options. These options are described below in the following functional groups: input packet selection, output file options, output visualization options, graphical options, and debugging.

Input Packet Selection

These options determine the packets that will be selected by **dnspktflow**. Short versions of the options are given in parentheses.

–ignore-hosts=STRING (-i)

A regular expression of host names to ignore in the query/response fields.

–only-hosts=STRING (-r)

A regular expression of host names to analyze in the query/response fields.

–show-frame-num (-f)

Display the packet frame numbers.

–begin-frame=INTEGER (-b)

Begin at packet frame NUMBER.

Output File Options

These options determine the type and location of **dnspktflow**'s output.

–output-file=STRING (-o)

Output file name (default: out

–fig

Output format should be fig.

–tshark-out=STRING (-O)

Save **tshark** output to this file.

–multiple-outputs (-m)

One picture per request (use

–magic-point=STRING (-M)

Saves a MagicPoint presentation for the output.

Output Visualization Options:

These options determine specifics of **dnspktflow**'s output.

–last-line-labels-only (-L)

Only show data on the last line drawn.

–most-lines=INTEGER (-z)

Only show at most INTEGER connections.

–input-is-tshark-out (-T)

The input file is already processed by **tshark**.

Graphical Options:

These options determine fields included in **dnspktflow**'s output.

–show-type (-t)

Shows message type in result image.

–show-queries (-q)

Shows query questions in result image.

–show-answers (-a)

Shows query answers in result image.

–show-authoritative (-A)

Shows authoritative information in result image.

–show-additional (-x)

Shows additional information in result image.

–show-label-lines (-l)

Shows lines attaching labels to lines.

–fontsize=INTEGER

Font Size

Debugging:

These options may assist in debugging **dnspktflow**.

–dump-pkts (-d)

Dump data collected from the packets.

–help (-h)

Show help for command line options.

SEE ALSO

Getopt::GUI::Long(3) Net::DNS(3) QWizard.pm(3)

2.2.2 donuts

NAME

donuts - Analyze DNS zone files for errors and warnings

SYNOPSIS

```
donuts [-h] [-H] [-v] [-l LEVEL] [-r RULEFILES] [-i IGNORELIST]
       [-C] [-c configfile] ZONEFILE DOMAINNAME...
```

DESCRIPTION

donuts is a DNS lint application that examines DNS zone files looking for particular problems. This is especially important for zones making use of DNSSEC security records, since many subtle problems can occur.

If the **Text::Wrap** Perl module is installed, **donuts** will give better output formatting.

OPTIONS

-h

Displays a help message.

-v

Turns on more verbose output.

-q

Turns on more quiet output.

-l *LEVEL*

Sets the level of errors to be displayed. The default is level 5. The maximum value is level 9, which displays many debugging results. You probably want to run no higher than level 8.

-r *RULEFILES*

A comma-separated list of rule files to load. The strings will be passed to *glob()* so wildcards can be used to specify multiple files.

-i *IGNORELIST*

A comma-separated list of regex patterns which are checked against rule names to determine if some should be ignored. Run with *-v* to figure out rule names if you're not sure which rule is generating errors you don't wish to see.

-L

Include rules that require live queries of data. Generally, these rules concentrate on pulling remote DNS data to test; for example, parent/child zone relationships.

-c *CONFIGFILE*

Parse a configuration file to change constraints specified by rules. This defaults to **$HOME/.donuts.conf**.

-C

Don't read user configuration files at all, such as those specified by the *-c* option or the **$HOME/.donuts.conf** file.

-t *INTERFACE*

Specifies that **tcpdump** should be started on *INTERFACE* (e.g., "eth0") just before donuts begins its run of rules for each domain and will stop it just after it has processed the rules. This is useful when you wish to capture the traffic generated by the *live* feature, described above.

-T *FILTER*

When **tcpdump** is run, this *FILTER* is passed to it for purposes of filtering traffic. By default, this is set to I<port 53 —— ip[6:2] & 0x1fff != 0>, which limits the traffic to traffic destined to port 53 (DNS) or fragmented packets.

-o *FILE*

Saves the **tcpdump** captured packets to *FILE*. The following special fields can be used to help generate unique file names:

This is replaced with the current domain name being analyzed (e.g., "example.com").

This is replaced with the current epoch time (i.e., the number of seconds since Jan 1, 1970).

This field defaults to **%d.%t.pcap**.

-H

Displays the personal configuration file rules and tokens that are acceptable in a configuration file. The output will consist of a rule name, a token, and a description of its meaning.

Your configuration file (e.g., **$HOME/.donuts.conf**) may have lines in it that look like this:

```
# change the default minimum number of legal NS records from 2 to 1
name: DNS_MULTIPLE_NS
minnsrecords: 1

# change the level of the following rule from 8 to 5
name: DNS_REASONABLE_TTLS
level: 5
```

This allows you to override certain aspects of how rules are executed.

-R

Displays a list of all known rules along with their description (if available).

-F LIST

–features=LIST

The *–features* option specifies additional rule features that should be executed. Some rules are turned off by default because they are more intensive or require a live network connection, for instance. Use the *–features* flag to turn them on. The LIST argument should be a comma separated list. Example usage:

```
--features live,data_check
```

Features available in the default rule set:

live

The *live* feature allows rules that need to perform live DNS queries to run. Most of these *live* rules query parent and children of the current zone, when appropriate, to see that the parent/child relationships have been built properly. For example, if you have a DS record which authenticates the key used in a child zone the *live* feature will let a rule run which checks to see if the child is actually publishing the DNSKEY that corresponds to the test zone's DS record.

–show-gui

[alpha code]

Displays a browsable GUI screen showing the results of the donuts tests.

The **QWizard** and **Gtk2** Perl modules must be installed for this to work.

–live

Obsolete command line option. Please use *–features live* instead.

SEE ALSO

For writing rules that can be loaded by donuts:
Net::DNS::SEC::Tools::Donuts::Rule

General DNS and DNSSEC usage:
Net::DNS(3), Net::DNS::SEC(3)

Gtk2.pm(3), QWizard.pm(3)

2.2.3 donutsd

NAME

donutsd - Run the **donuts** syntax checker periodically and report the results to an administrator

SYNOPSIS

```
    donutsd [-z FREQ] [-t TMPDIR] [-f FROM] [-s SMTPSERVER] [-a DONUTSARGS]
            [-x] [-v] [-i zonelistfile] [ZONEFILE ZONENAME ZONECONTACT]
```

DESCRIPTION

donutsd runs **donuts** on a set of zone files every so often (the frequency is specified by the *-z* flag which defaults to 24 hours) and watches for changes in the results. These changes may be due to the time-sensitive nature of DNSSEC-related records (e.g., RRSIG validity periods) or because parent/child relationships have changed. If any changes have occurred in the output since the last run of **donuts** on a particular zone file, the results are emailed to the specified zone administrator's email address.

OPTIONS

-v

Turns on more verbose output.

-o

Run once and quit, as opposed to sleeping or re-running forever.

-a ARGUMENTS

Passes arguments to command line arguments of **donuts** runs.

-z TIME

Sleeps TIME seconds between calls to **donuts**.

-e ADDRESS

Mail ADDRESS with a summary of the results from all the files. These are the last few lines of the **donuts** output for each zone that details the number of errors found.

-s SMTPSERVER

When sending mail, send it to the SMTPSERVER specified. The default is *localhost*.

-f FROMADDR

When sending mail, use FROMADDR for the From: address.

-x

Send the **diff** output in the email message as well as the **donuts** output.

-t TMPDIR

Store temporary files in TMPDIR.

-i INPUTZONES

See the next section details.

ZONE ARGUMENTS

The rest of the arguments to **donutsd** should be triplets of the following information:

ZONEFILE

The zone file to examine.

ZONENAME

The zonename that file is supposed to be defining.

ZONECONTACT

An email address of the zone administrator (or a comma-separated list of addresses.) The results will be sent to this email address.

Additionally, instead of listing all the zones you wish to monitor on the command line, you can use the *-i* flag which specifies a file to be read listing the TRIPLES instead. Each line in this file should contain one triple with white-space separating the arguments.

Example:

```
    db.zonefile1.com   zone1.com   admin@zone1.com
    db.zonefile2.com   zone2.com   admin@zone2.com,admin2@zone2.com
```

For even more control, you can specify an XML file (whose name must end in **.xml**) that describes the same information. This also allows for per-zone customization of the **donuts** arguments. The **XML::Smart** Perl module must be installed in order to use this feature.

```
    <donutsd>
       <zones>
          <zone>
              <file>db.example.com</file>
              <name>example.com</name>
              <contact>admin@example.com</contact>
              <!-- this is not a signed zone therefore we'll
                   add these args so we don't display DNSSEC errors -->
              <donutsargs>-i DNSSEC</donutsargs>
          </zone>
       </zones>
    </donutsd>
```

The **donutsd** tree may also contain a *configs* section where command-line flags can be specified:

```
<donutsd>
    <configs>
        <config><flag>a</flag><value>--live --level 8</value></config>
        <config><flag>e</flag><value>wes@example.com</value></config>
    </configs>
    <zones>
         ...
    </zones>
</donutsd>
```

Real command line flags will be used in preference to those specified in the **.xml** file, however.

EXAMPLE

```
donutsd -a "--live --level 8" -f root@somewhere.com $\$
   db.example.com example.com admin@example.com
```

SEE ALSO

donuts(8)

2.2.4 getaddr

NAME

getaddr - command-line test program for the *val_gettaddrinfo()* function

SYNOPSIS

```
getaddr [options] <hostname|IPv4 address|IPv6 address>
```

DESCRIPTION

This utility is a command-line wrapper around the *val_getaddrinfo()* function. It invokes the *val_getaddrinfo()* operation for the given command-line arguments and displays the return code from the function and the contents of the *val_addrinfo* structure.

The exit status for this program is 1 if the answer received is trusted, 2 if the answer received is validated, and -1 for an error. The trusted and validated status values are obtained using the *val_istrusted()* and *val_isvalidated()* functions from *libval(3)*.

OPTIONS

-n, –novalidate

Do not perform validation.

-c, –canonname

Use the AI_CANONNAME flag in *val_getaddrinfo()*.

-s, –service=*PORT—SERVICE*

Use the specified transport-layer port or service name.

-h, –help

Display the help and exit.

PRE-REQUISITES

libval(3)

SEE ALSO

libval(3), val_getaddrinfo(3)

2.2.5 getdnskeys

NAME

getdnskeys - Manage lists of DNSKEYs from DNS zones

SYNOPSIS

```
getdnskeys [-i file] [-o file] [-k] [-T] [-t] [-v] [zones]
```

DESCRIPTION

getdnskeys manages lists of DNSKEYs from DNS zones. It may be used to retrieve and compare DNSKEYs. The output from **getdnskeys** may be included (directly or indirectly) in a **named.conf** file.

OPTIONS

-h

Gives a help message.

-i path

Reads *path* as a **named.conf** with which to compare key lists.

-k

Only looks for Key Signing Keys (KSKs); all other keys are ignored.

-o file

Writes the results to *file*.

-T

Checks the current trusted key list from **named.conf**.

-t

Encloses output in needed **named.conf** syntax markers.

-v

Turns on verbose mode for additional output.

EXAMPLES

This **getdnskeys** will retrieve the KSK for example.com:

```
getdnskeys -o /etc/named.trustkeys.conf -k -v -t example.com
```

This **getdnskeys** will check saved keys against a live set of keys:

```
    getdnskeys -i /etc/named.trustkeys.conf -T -k -v -t
```

This **getdnskeys** will automatically update a set of saved keys:

```
    getdnskeys -i /etc/named.trustkeys.conf -k -t -T -v
               -o /etc/named.trustkeys.conf
```

SECURITY ISSUES

Currently this does not validate new keys placed in the file in any way, nor does it validate change over keys which have been added.

It also does not handle revocation of keys.

It should prompt you before adding a new key so that you can always run the auto-update feature.

2.2.6 gethost

NAME

gethost - command-line test program for the *val_gethostbyname()* (and related) functions

SYNOPSIS

```
gethost [options] name
```

DESCRIPTION

This utility is a command-line wrapper around the *val_gethostbyname()* (and related) functions. It invokes the *val_gethostbyname()*, *val_gethostbyname_r()* and *val_gethostbyname2()* functions for the given command-line arguments and displays the returned *hostent*, the validation status value, and the value of the *h_errno* variable.

The exit status for this program is 1 if the answer received is trusted, 2 if the answer received is validated, and -1 for an error. The trusted and validated status values are obtained using the *val_istrusted()* and *val_isvalidated()* functions from *libval(3)*.

OPTIONS

-n, –novalidate

Do not perform validation.

-r, –reentrant

Use the re-entrant query interface.

-f, –family=*[AF_INET—AF_INET6]*

Use the specified address family for the query.

-h, –help

Display the help and exit.

PRE-REQUISITES

libval

SEE ALSO

libval(3), val_getaddrinfo(3)

2.2.7 libval_check_conf

NAME

libval_check_conf - command-line program for checking validity of the validator configuration file

SYNOPSIS

```
libval_check_conf [options] <dnsval.conf>
```

DESCRIPTION

This program checks the validity of a given validator configuration file, typically named **dnsval.conf**, for any syntax errors. The input to this program is a validator configuration file, but the program also requires valid **resolv.conf** and **root.hints** to be specified. The location for these auxiliary files may be specified using the *-r* and *-i* options respectively. If these options are not provided, the default **resolv.conf** and **root.hints** files recognized by the validator library are used instead. The program displays an error if the default **resolv.conf** and **root.hints** files, and the *-i* and *-r* are both unspecified.

RETURN VALUES

The program returns 0 on success and -1 on failure.

OPTIONS

-r, –resolv-conf

Specifies the location for the **resolv.conf** file. If this option is not specified the default **resolv.conf** file recognized by the validator library is used instead.

-i, –root-hints

Specifies the location for the **root.hints** file. If this option is not specified the default **root.hints** file recognized by the validator library is used instead.

-v, –verbose

Displays detailed messages and warnings associated with the validator configuration file check process.

-h, –help

Displays a usage help message and exits the program.

SEE ALSO

dnsval.conf(3)

2.2.8 mapper

NAME

mapper - Create graphical maps of DNS zone data

SYNOPSIS

```
    mapper [options] zonefile1 ... zonefileN
```

DESCRIPTION

This application creates a graphical map of one or more zone files. The output gives a graphical representation of a DNS zone or zones. The output is written in the PNG format. The result can be useful for getting a more intuitive view of a zone or set of zones. It is extremely useful for visualizing DNSSEC deployment within a given zone as well as to help discover problem spots.

OPTIONS

-h

Prints a help summary.

-o OUTFILE.png

Saves the results to a given filename. If this option is not given, the map will be saved to **map.png**.

-r

Lists resource records assigned to each node within the map.

-t TYPE,TYPE...

Adds the data portion of a resource record to the displayed node information. Data types passed will be automatically converted to upper-case for ease of use.

Example usage: *-t A* will add IPv4 addresses to all displayed nodes that have A records.

-L

Adds a legend to the map.

-l (neato—dot—twopi—circo—fdp)

Selects a layout format. The default is *neato*, which is circular in pattern. See the documentation on the **GraphViz** package and the **GraphViz** Perl module for further details.

-a

Allows overlapping of nodes. This makes much tighter maps with the downside being that they are somewhat cluttered. Maps of extremely large zones will be difficult to decipher if this option is not used.

-e WEIGHT

Assigns an edge weight to edges. In theory, >1 means shorter and <1 means longer, although, it may not have any effect as implemented. This should work better in the future.

-f INTEGER

Uses the INTEGER value for the font size to print node names with. The default value is 10.

-w WARNTIME

Specifies how far in advance expiration warnings are enabled for signed resource records. The default is 7 days. The warning time is measured in seconds.

-i REGEX

Ignores record types matching a *REGEX* regular expression.

-s TYPE,TYPE...

Specifies a list of record types that will not be analyzed or displayed in the map. By default, this is set to NSEC and CNAME in order to reduce clutter. Setting it to "" will display these results again.

-T TYPE,TYPE...

Restrict record types that will be processed to those of type *TYPE*. This is the converse of the *-s* option. It is not meaningful to use both *-s* and *-t* in the same invocation. They will both work at once, however, so if *-T* specifies a type which *-s* excludes, it will not be shown.

-g

Attempts to cluster nodes around the domain name. For "dot" layouts, this actually means drawing a box around the cluster. For the other types, it makes very little difference, if any.

-q

Prevents output of warnings or errors about records that have DNSSEC signatures that are near or beyond their signature lifetimes.

EXAMPLE INVOCATIONS

mapper -s cname,nsec -i dhcp -L zonefile zone.com

Writes to the default file (**map.png**) of a *zone.com* zone stored in **zonefile**. It excludes any hosts with a name containing **dhcp** and ignores any record of type *CNAME* or *NSEC*. A legend is included in the output.

mapper -s txt,hinfo,cname,nsec,a,aaaa,mx,rrsig -L zonefile zone.com zonefile2 sub.zone.com ...

Removes a lot of records from the display in order to primarily display a map of a zone hierarchy.

mapper -l dot -s txt,hinfo,cname,nsec,a,aaaa,mx,rrsig -L zonefile zone.com zonefile2 sub.zone.com ...

As the previous example, but this command draws a more vertical tree-style graph of the zone. This works well for fairly deep but narrow hierarchies. Tree-style diagrams rarely look as nice for full zones.

SEE ALSO

Net::DNS(3)

2.2.9 tachk

NAME

tachk - Check the validity of the trust anchors in a **named.conf** file

SYNOPSIS

```
  tachk [options] <named.conf>
```

DESCRIPTION

tachk checks the validity of the trust anchors in the specified **named.conf** file. The output given depends on the options selected.

Note: This script may be removed in future releases.

OPTIONS

tachk takes two types of options: record-attribute options and output-style options. These option sets are detailed below.

Record-Attribute Options

These options define which trust anchor records will be displayed.

-valid

This option displays the valid trust anchors in a **named.conf** file.

-invalid

This option displays the invalid trust anchors in a **named.conf** file.

Output-Format Options

These options define how the trust anchor information will be displayed. Without any of these options, the zone name and key tag will be displayed for each trust anchor.

-count

The count of matching records will be displayed, but the matching records will not be.

-long

The long form of output will be given. The zone name and key tag will be displayed for each trust anchor.

-terse

This option displays only the name of the zones selected by other options.

-help

Display a usage message.

SEE ALSO

trustman(8)

2.2.10 validate

NAME

validate - Query the Domain Name System and display results of the DNSSEC validation process

SYNOPSIS

```
    validate

    validate [options] DOMAIN_NAME
```

DESCRIPTION

validate is a diagnostic tool which uses the DNSSEC validator. It takes *DOMAIN_NAME* as an argument and queries the DNS for that domain name. It outputs the series of responses that were received from the DNS and the DNSSEC validation results for each domain name. An examination of the queries and validation results can help an administrator uncover errors in DNSSEC configuration of DNS zones.

If no options are specified and no *DOMAIN_NAME* argument is given, **validate** will perform a series of pre-defined test queries against the *test.dnssec-tools.org* zone. This serves as a test-suite for the validator. If any options are specified (e.g., configuration file locations), *-s* or *–selftest* must be specified to run the test-suite.

OPTIONS

-c *CLASS*, –class=*CLASS*

This option can be used to specify the DNS class of the Resource Record queried. If this option is not given, the default class **IN** is used.

-h, –help

Display the help and exit.

-m, –merge

When this option is given, **validate** will merge different RRsets in the response into a single answer. If this option is not given, each RRset is output as a separate response. This option makes sense only when used with the *-p* option.

-p, –print

Print the answers and validation results. By default, **validate** just outputs a series of responses and their validation results on *stderr*. When the *-p* option is used, **validate** will also output the final result on *stdout*.

-t *TYPE*, –type=*TYPE*

This option can be used to specify the DNS type of the Resource Record queried. If this option is not given, **validate** will query for the **A** record for the given *DOMAIN_NAME*.

-v *FILE*, –dnsval-conf=*FILE*

This option can be used to specify the location of the **dnsval.conf** configuration file.

-r *FILE*, –resolv-conf=*FILE*

This option can be used to specify the location of the **resolv.conf** configuration file containing the name servers to use for lookups.

-i *FILE*, –root-hints=*FILE*

This option can be used to specify the location of the root.hints configuration file, containing the root name servers. This is only used when no name server is found, and **validate** must do recursive lookups itself.

-S *suite*[:*suite*], –test-suite=*suite*[:*suite*]

This option specifies the test suite (or range of test suites) to use for the internal tests.

-s, –selftest

This option can be used to specify that the application should perform its test-suite against the *dnssec-tools.org* test domain. If the name servers configured in the system **resolv.conf** do not support DNSSEC, use the *-r* and *-i* options to enable **validate** to use its own internal recursive resolver.

-T *number*[:*number*], –testcase=*number*[:*number*]

This option can be used to run a specific test (or range of tests) from the test suite.

-F *file*, –testcase-conf=*file*

This option is used to specify the file containing the test cases.

-l *label*, –label=*label*

This option can be used to specify the policy from within the *dnsval.conf* file to use during validation.

-w *seconds*, –wait=*seconds*

This option can be used to run the queries specified by other flags in a loop, with the specified interval between successive queries.

-o, –output=<debug-level>:<dest-type>[:<dest-options>]

<debug-level> is 1-7, corresponding to syslog levels ALERT-DEBUG
<dest-type> is one of file, net, syslog, stderr, stdout
<dest-options> depends on <dest-type>

```
file:<file-name>   (opened in append mode)
net[:<host-name>:<host-port>] (127.0.0.1:1053
syslog[:facility] (0-23 (default 1 USER))
```

PRE-REQUISITES

libval

SEE ALSO

libval(3), syslog(3)

2.3 Zone-Signing Commands

The zone-signing commands provide tools to assist in the signing DNS zone files and keeping records about those signed zones. These commands are:

zonesigner	generates encryption keys and signs a DNS zone
genkrf	creates a new *keyrec* file
krfcheck	checks a *keyrec* file for errors
lskrf	lists the contents of a *keyrec* file
expchk	checks a *keyrec* file for expired zones
fixkrf	fixes a *keyrec* file for missing keys
cleanarch	cleans a key archive directory
cleankrf	cleans a *keyrec* file of orphaned keys
signset-editor	GUI editor for signing sets in a *keyrec* file

2.3.1 zonesigner

NAME

zonesigner - Generates encryption keys and signs a DNS zone

SYNOPSIS

```
zonesigner [options] <zone-file> <signed-zone-file>

# get started immediately examples:

# first run on a zone for example.com:
zonesigner -genkeys -endtime +2678400 example.com

# future runs before expiration time (reuses the same keys):
zonesigner -endtime +2678400 example.com
```

DESCRIPTION

This script combines into a single command many actions that are required to sign a DNS zone. It generates the required KSK and ZSK keys, adds the key data to a zone record file, signs the zone file, and runs checks to ensure that everything worked properly. It also keeps records about the keys and how the zone was signed in order to facilitate re-signing of the zone in the future.

The **zonesigner**-specific zone-signing records are kept in *keyrec* files. Using *keyrec* files, defined and maintained by DNSSEC-Tools, **zonesigner** can automatically gather many of the options used to previously sign and generate a zone and its keys. This allows the zone to be maintained using the same key lengths and expiration times, for example, without an administrator needing to manually track these fields.

QUICK START

The following are examples that will allow a quick start on using **zonesigner**:

first run on example.com

The following command will generate keys and sign the zone file for example.com, giving an expiration date 31 days in the future. The zone file is named **example.com** and the signed zone file will be named **example.com.signed**.

```
zonesigner -genkeys -endtime +2678400 example.com
```

subsequent runs on example.com

The following command will re-sign example.com's zone file, but will not generate new keys. The files and all key-generation and zone-signing arguments will remain the same.

```
        zonesigner example.com
```

USING ZONESIGNER

zonesigner is used in this way:

```
    zonesigner [options] <zone-file> <signed-zone-file>
```

The *zone-file* argument is required.

zone-file is the name of the zone file from which a signed zone file will be created. If the *-zone* option is not given, then *zone-file* will be used as the name of the zone that will be signed. Generated keys are given this name as their base.

The zone file is modified to have **include** commands, which will include the KSK and ZSK keys. These lines are placed at the end of the file and should not be modified by the user. If the zone file already includes any key files, those inclusions will be deleted. These lines are distinguished by starting with "$INCLUDE" and end with **.key**. Only the actual include lines are deleted; any related comment lines are left untouched.

An intermediate file is used in signing the zone. *zone-file* is copied to the intermediate file and is modified in preparation of signing the zone file. Several $INCLUDE lines will be added at the end of the file and the SOA serial number will be incremented.

signed-zone is the name of the signed zone file. If it is not given on the command line, the default signed zone filename is the *zone-file* appended with **.signed**. Thus, executing **zonesigner example.com** will result in the signed zone being stored in **example.com.signed**.

Unless the *-genkeys*, *-genksk*, *-genzsk*, or *-newpubksk* options are specified, the last keys generated for a particular zone will be used in subsequent **zonesigner** executions.

KEYREC FILES

keyrec files retain information about previous key-generation and zone-signing operations. If a *keyrec* file is not specified (by way of the *-krfile* option), then a default *keyrec* file is used. If this default is not specified in the system's DNSSEC-Tools configuration file, the filename will be the zone name appended with **.krf**. If the *-nokrfile* option is given, then no *keyrec* file will be consulted or saved.

keyrec files contain three types of entries: zone *keyrec*s, set *keyrec*s, and key *keyrec*s. Zone *keyrec*s contain information specifically about the zone, such as the number of ZSKs used to sign the zone, the end-time for the zone, and the key signing set names (names of set *keyrec*s.) Set *keyrec*s contain lists of key *keyrec* names used for a specific purpose, such as the current ZSK keys or the published ZSK keys. Key *keyrec*s contain information about the generated keys themselves, such as encryption algorithm, key length, and key lifetime.

Each *keyrec* contains a set of "key/value" entries, one per line. Example 4 below contains the contents of a sample *keyrec* file.

ENTROPY

On some systems, the implementation of the pseudo-random number generator requires keyboard activity. This keyboard activity is used to fill a buffer in the system's random number generator. If **zonesigner** appears hung, you may have to add entropy to the random number generator by randomly striking keys until the program completes. Display of this message is controlled by the **entropy_msg** configuration file parameter.

DETERMINING OPTION VALUES

zonesigner checks four places in order to determine option values. In descending order of precedence, these places are:

command line options

keyrec file

DNSSEC-Tools configuration file

zonesigner defaults

Each is checked until a value is found. That value is then used for that **zonesigner** execution and the value is stored in the *keyrec* file.

Example

For example, the KSK length has the following values:

-ksklength command line option	8192
keyrec file	1024
DNSSEC-Tools configuration file	2048
zonesigner defaults	512

If all are present, then the KSK length will be 8192.

If the *-ksklength* command line option wasn't given, the KSK length will be 1024.

If the KSK length wasn't given in the configuration file, it will be 8192.

If the KSK length wasn't in the *keyrec* file or the configuration file, the KSK length will be 8192.

If the *-ksklength* command line option wasn't given and the KSK length wasn't in the configuration file, it'll be 1024.

If the command line option wasn't given, the KSK length wasn't in the *keyrec* file, and it wasn't in the configuration file, then the KSK length will be 512.

OPTIONS

Three types of options may be given, based on the command for which they are intended. These commands are **dnssec-keygen**, **dnssec-signzone**, and **zonesigner**.

zonesigner-specific Options

-nokrfile

No *keyrec* file will be consulted or created.

-krfile

keyrec file to use in processing options. See the man page for the DNSSEC-Tools **tooloptions.pm** module for more details about this file.

-genkeys

Generate new KSKs and ZSKs for the zone.

-genksk

Generate new Current KSKs for the zone. Any existing Current KSKs will be marked as obsolete. If this option is not given, the last KSKs generated for this zone will be used.

-genzsk

Generate new ZSKs for the zone. By default, the last ZSKs generated for this zone will be used.

-newpubksk

Generate new Published KSKs for the zone. Any existing Published KSKs will be marked as obsolete.

-useboth

Use the existing Current **and** Published ZSKs to sign the zone.

-usezskpub

Use the existing Published ZSKs to sign the zone.

-archivedir

The key archive directory. If a key archive directory hasn't been specified (on the command line or in the DNSSEC-Tools configuration file) and the *-nosave* option was **not** given, an error message will be displayed and **zonesigner** will exit.

When the files are saved into the archive directory, the existing file names are prepended with a timestamp. The timestamp indicates when the files are archived.

This directory **may not** be the root directory.

-nosave

Do not save obsolete keys to the key archive directory. The default behavior is to save obsolete keys.

-kskcount

The number of KSK keys to generate and with which to sign the zone. The default is to use a single KSK key.

-ksklife

The time between KSK rollovers. This is measured in seconds.

-ksignset

The name of the KSK signing set to use. If the signing set does not exist, then this must be used in conjunction with either *-genkeys* or *-genksk*. The name may contain alphanumerics, underscores, hyphens, periods, and commas.

The default signing set name is "signing-set-*N*", where *N* is a number. If *-signset* is not specified, then **zonesigner** will use the default and increment the number for subsequent signing sets.

-zsklife

The time between ZSK rollovers. This is measured in seconds.

-zskcount

The number of ZSK keys to generate and with which to sign the zone. The default is to use a single ZSK key.

-signset

The name of the ZSK signing set to use as the Current ZSK signing set. The zone is signed and the given signing set becomes the zone's new Current ZSK signing set. If the signing set does not exist, then this must be used in conjunction with either *-genkeys* or *-genzsk*.

The name may contain alphanumerics, underscores, hyphens, periods, and commas. The default signing set name is "signing-set-*N*", where *N* is a number. If *-signset* is not specified, then **zonesigner** will use the default and increment the number for subsequent signing sets.

-rollksk

Force a rollover of the KSK keys. The Current KSK keys are marked as Obsolete and the Published KSK keys are marked as Current. The zone is then signed with the new set of Current KSK keys. If the zone's *keyrec* does not list a Current or Published KSK, an error message is printed and **zonesigner** stops execution.

The zone's *keyrec* file is updated to show the new key state.

The *keyrec*s of the KSK keys are adjusted as follows:

1. The Current KSK keys are marked as Obsolete.
2. The Published KSK keys are marked as Current.
3. The obsolete KSK keys are moved to the archive directory.

Warning: The timing of key-rolling is critical. Great care must be taken when using this option. **rollerd** automates the KSK rollover process and may be used to safely take care of this aspect of DNSSEC management.

Warning: Using the *-rollksk* option should only be used if you know what you're doing.

Warning: This is a *temporary* method of KSK rollover. It *may* be changed in the future.

-rollzsk

Force a rollover of the ZSK keys using the Pre-Publish Key Rollover method. The rollover process adjusts the keys used to sign the specified zone, generates new keys, signs the zone with the appropriate keys, and updates the *keyrec* file. The Pre-Publish Key Rollover process is described in the DNSSEC Operational Practices document.

Three sets of ZSK keys are used in the rollover process: Current, Published, and New. Current ZSKs are those which are used to sign the zone. Published ZSKs are available in the zone data, and therefore in cached zone data, but are not yet used to sign the zone. New ZSKs are not available in zone data nor yet used to sign the zone, but are waiting in the wings for future use.

The *keyrec*s of the ZSK keys are adjusted as follows:

1. The Current ZSK keys are marked as obsolete.
2. The Published ZSK keys are marked as Current.
3. The New ZSK keys, if they exist, are marked as Published.
4. Another set of ZSK keys are generated, which will be marked as the New ZSK keys.
5. The Published ZSK keys' zsklife field is copied to the new ZSK keys' keyrecs.
6. The obsolete ZSK keys are moved to the archive directory.

The quick summary of proper ZSK rolling (which **rollerd** does for you if you use it):

1. wait 2 * max(TTL in zone)
2. run zonesigner using -usezskpub
3. wait 2 * max(TTL in zone)
4. run zonesigner using -rollzsk
5. wait 2 * max(TTL in zone)

Warning: The timing of key-rolling is critical. Great care must be taken when using this option. **rollerd** automates the rollover process and may be used to safely take care of this aspect of DNSSEC management. Using the *-rollzsk* option should only be used if you know what you're doing.

-intermediate

Filename to use for the temporary zone file. The zone file will be copied to this file and then the key names appended.

-zone

Name of the zone that will be signed. This zone name may be given with this option or as the first non-option command line argument.

-help

Display a usage message.

-Version

Display the version information for **zonesigner** and the DNSSEC-Tools package.

-verbose

Verbose output will be given. As more instances of *-verbose* are given on the command line, additional levels of verbosity are achieved.

Verbosity Level	Output
1	operations being performed (e.g., generating key files, signing zone)
2	details on operations and some operation results (e.g., new key names, zone serial number)
3	operations' parameters and additional details (e.g., key lengths, encryption algorithm, executed commands)

Table 2: **zonesigner** Verbosity Levels

Higher levels of verbosity are cumulative. Specifying two instances of *-verbose* will get the output from the first and second levels of output.

-showkeycmd

Display the actual key-generation command (with options and arguments) that is executed. This is a small subset of verbose level 3 output.

-showsigncmd

Display the actual zone-signing command (with options and arguments) that is executed. This is a small subset of verbose level 3 output.

dnssec-keygen-specific Options

-algorithm

Cryptographic algorithm used to generate the zone's keys. The default value is RSASHA1. The option value is passed to **dnssec-keygen** as the the *-a* flag. Consult **dnssec-keygen**'s manual page to determine legal values.

-ksklength

Bit length of the zone's KSK key. The default is 1024.

-random

Source of randomness used to generate the zone's keys. This is assumed to be a file, for example **/dev/urandom**.

-zsklength

Bit length of the zone's ZSK key. The default is 512.

-kgopts

Additional options for **dnssec-keygen** may be specified using this option. The additional options are passed as a single string value as an argument to the *-kgopts* option.

dnssec-signzone-specific Options

-endtime

Time that the zone expires, measured in seconds. See the man page for **dnssec-signzone** for the valid format of this field. The default value is 2592000 seconds (30 days.)

-gends

Force **dnssec-signzone** to generate DS records for the zone. This option is translated into *-g* when passed to **dnssec-signzone**.

-ksdir

Specify a directory for storing keysets. This is passed to **dnssec-signzone** as the *-d* option.

-szopts

Additional options for **dnssec-signzone** may be specified using this option. The additional options are passed as a single string value as an argument to the *-szopts* option.

Other Options

-zcopts

Additional options for **named-checkzone** may be specified using this option. The additional options are passed as a single string value as an argument to the *-zcopts* option.

Examples

Example 1.

In the first example, an existing *keyrec* file is used to assist in signing the example.com domain. Zone data are stored in **example.com**, and the keyrec is in **example.krf**. The final signed zone file will be **db.example.com.signed**. Using this execution:

```
    # zonesigner -krfile example.krf example.com db.example.com.signed
```

the following files are created:

Kexample.com.+005+45842.private
Kexample.com.+005+45842.key
Kexample.com.+005+50186.private
Kexample.com.+005+50186.key
Kexample.com.+005+59143.private
Kexample.com.+005+59143.key
dsset-example.com.
keyset-example.com.
db.example.com.signed

The first six files are the KSK and ZSK keys required for the zone. The next two files are created by the zone-signing process. The last file is the final signed zone file.

Example 2.

In the second example, an existing *keyrec* file is used to assist in signing the example.com domain. Zone data are stored in **example.com**, and the keyrec is in **example.krf**. The generated keys, an intermediate zone file, and final signed zone file will use **example.com** as a base. Using this execution:

```
# zonesigner -krfile example.krf -intermediate example.zs example.com
```

the following files are created:

Kdb.example.com.+005+12354.key
Kdb.example.com.+005+12354.private
Kdb.example.com.+005+82197.key
Kdb.example.com.+005+82197.private
Kdb.example.com.+005+55888.key
Kdb.example.com.+005+55888.private
dsset-db.example.com.
keyset-db.example.com.
example.zs
example.com.signed

The first six files are the KSK and ZSK keys required for the zone. The next two files are created by the zone-signing process. The second last file is an intermediate file that will be signed. The last file is file is the final signed zone.

Example 3.

In the third example, no *keyrec* file is specified for the signing of the example.com domain. In addition to files created as shown in previous examples, a new *keyrec* file is created. The new *keyrec* file uses the domain name as its base. Using this execution:

```
# zonesigner example.com db.example.com
```

The *keyrec* file is created as **example.com.krf**.

The signed zone file is created in **db.example.com**.

Example 4.

This example shows a *keyrec* file generated by **zonesigner**.

The command executed is:

```
# zonesigner example.com db.example.com
```

The generated *keyrec* file contains six *keyrec*s: a zone *keyrec*, two set *keyrec*s, one KSK *keyrec*, and two ZSK *keyrec*s.

```
zone        "example.com"
    zonefile        "example.com"
    signedzone      "db.example.com"
    endtime         "+2592000"
    kskcur          "signing-set-24"
    kskdirectory    "."
    zskcur          "signing-set-42"
    zskpub          "signing-set-43"
    zskdirectory    "."
    keyrec_type     "zone"
    keyrec_signsecs "1115166642"
    keyrec_signdate "Wed May  4 00:30:42 2005"

set                "signing-set-24"
    zonename        "example.com"
    keys            "Kexample.com.+005+24082"
    keyrec_setsecs  "1110000042"
    keyrec_setdate  "Sat Mar  5 05:20:42 2005"

set                "signing-set-42"
    zonename        "example.com"
    keys            "Kexample.com.+005+53135"
    keyrec_setsecs  "1115166640"
    keyrec_setdate  "Wed May  4 00:30:40 2005"

set                "signing-set-43"
    zonename        "example.com"
    keys            "Kexample.com.+005+13531"
    keyrec_setsecs  "1115166641"
    keyrec_setdate  "Wed May  4 00:30:41 2005"

key                "Kexample.com.+005+24082"
    zonename        "example.com"
    keyrec_type     "kskcur"
    algorithm       "rsasha1"
```

```
        random          "/dev/urandom"
        keypath         "./Kexample.com.+005+24082.key"
        ksklength       "1024"
        ksklife         "15768000"
        keyrec_gensecs  "1110000042"
        keyrec_gendate  "Sat Mar  5 05:20:42 2005"

    key                "Kexample.com.+005+53135"
        zonename        "example.com"
        keyrec_type     "zskcur"
        algorithm       "rsasha1"
        random          "/dev/urandom"
        keypath         "./Kexample.com.+005+53135.key"
        zsklength       "512"
        zsklife         "604800"
        keyrec_gensecs  "1115166638"
        keyrec_gendate  "Wed May  4 00:30:38 2005"

    key                "Kexample.com.+005+13531"
        zonename        "example.com"
        keyrec_type     "zskpub"
        algorithm       "rsasha1"
        random          "/dev/urandom"
        keypath         "./Kexample.com.+005+13531.key"
        zsklength       "512"
        zsklife         "604800"
        keyrec_gensecs  "1115166638"
        keyrec_gendate  "Wed May  4 00:30:38 2005"
```

NOTES

- One Zone in a *keyrec* File

  There is a bug in the signing-set code that necessitates only storing one zone in a *keyrec* file.

- SOA Serial Numbers

  Serial numbers in SOA records are merely incremented in this version. Future plans are to allow for more flexible serial number manipulation.

SEE ALSO

dnssec-keygen(8), dnssec-signzone(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3), Net::DNS::SEC::Tools::keyrec.pm(3), Net::DNS::SEC::Tools::tooloptions.pm(3)

2.3.2 genkrf

NAME

genkrf - Generate a *keyrec* file from Key Signing Key (KSK) and/or Zone Signing Key (ZSK) files

SYNOPSIS

```
  genkrf [options] <zone-file> [<signed-zone-file>]
```

DESCRIPTION

genkrf generates a *keyrec* file from KSK and/or ZSK files. It generates new KSK and ZSK keys if needed.

The name of the *keyrec* file to be generated is given by the *-krfile* option. If this option is not specified, **zone-name.krf** is used as the name of the *keyrec* file. If the *keyrec* file already exists, it will be overwritten with new *keyrec* definitions.

The *zone-file* argument is required. It specifies the name of the zone file from which the signed zone file was created. The optional *signed-zone-file* argument specifies the name of the signed zone file. If it is not given, then it defaults to **zone-file.signed**.

OPTIONS

genkrf has a number of options that assist in creation of the *keyrec* file. These options will be set to the first value found from this search path:

- command line options
- DNSSEC-Tools configuration file
- DNSSEC-Tools defaults

See **tooloptions.pm(3)** for more details. Exceptions to this are given in the option descriptions.

The **genkrf** options are described below.

General genkrf Options

-zone zone-name

This option specifies the name of the zone. If it is not given then *zone-file* will be used as the name of the zone.

-krfile keyrec-file

This option specifies the name of the *keyrec* file to be generated. If it is not given, then **zone-name.krf** will be used.

-algorithm algorithm

This option specifies the algorithm used to generate encryption keys.

-endtime endtime

This option specifies the time that the signature on the zone expires, measured in seconds.

-random random-device

Source of randomness used to generate the zone's keys. See the man page for **dnssec-signzone** for the valid format of this field.

-verbose

Display additional messages during processing. If this option is given at least once, then a message will be displayed indicating the successful generation of the *keyrec* file. If it is given twice, then the values of all options will also be displayed.

-help

Display a usage message.

KSK-related Options

-kskcur KSK-name

This option specifies the Current KSK's key file being used to sign the zone. If this option is not given, a new KSK will be created.

-kskcount KSK-count

This option specifies the number of KSK keys that will be generated. If this option is not given, the default given in the DNSSEC-Tools configuration file will be used.

-kskdir KSK-directory

This option specifies the absolute or relative path of the directory where the KSK resides. If this option is not given, it defaults to the current directory ".".

-ksklength KSK-length

This option specifies the length of the KSK encryption key.

-ksklife KSK-lifespan

This option specifies the lifespan of the KSK encryption key. This lifespan is **not** inherent to the key itself. It is **only** used to determine when the KSK must be rolled over.

ZSK-related Options

-zskcur ZSK-name

This option specifies the current ZSK being used to sign the zone. If this option is not given, a new ZSK will be created.

-zskpub ZSK-name

This option specifies the published ZSK for the zone. If this option is not given, a new ZSK will be created.

-zskcount ZSK-count

This option specifies the number of current and published ZSK keys that will be generated. If this option is not given, the default given in the DNSSEC-Tools configuration file will be used.

-zskdir ZSK-directory

This option specifies the absolute or relative path of the directory where the ZSKs reside. If this option is not given, it defaults to the current directory ".".

-zsklength ZSK-length

This option specifies the length of the ZSK encryption key.

-zsklife ZSK-lifespan

This option specifies the lifespan of the ZSK encryption key. This lifespan is **not** inherent to the key itself. It is **only** used to determine when the ZSK must be rolled over.

SEE ALSO

dnssec-keygen(8), dnssec-signzone(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3),
Net::DNS::SEC::Tools::keyrec.pm(3)

conf(5), keyrec(5)

2.3.3 krfcheck

NAME

krfcheck - Check a DNSSEC-Tools *keyrec* file for problems and inconsistencies

SYNOPSIS

```
  krfcheck [-zone | -set | -key] [-count] [-quiet]
           [-verbose] [-Version] [-help] keyrec-file
```

DESCRIPTION

This script checks a *keyrec* file for problems, potential problems, and inconsistencies.

Recognized problems include:

- no zones defined

  The *keyrec* file does not contain any zone *keyrec*s.

- no sets defined

  The *keyrec* file does not contain any set *keyrec*s.

- no keys defined

  The *keyrec* file does not contain any key *keyrec*s.

- unknown zone *keyrec*s

  A set *keyrec* or a key *keyrec* references a non-existent zone *keyrec*.

- missing key from zone *keyrec*

  A zone *keyrec* does not have both a KSK key and a ZSK key.

- missing key from set *keyrec*

  A key listed in a set *keyrec* does not have a key *keyrec*.

- expired zone *keyrec*s

  A zone has expired.

- mislabeled key

  A key is labeled as a KSK (or ZSK) and its owner zone has it labeled as the opposite.

- invalid zone data values

  A zone's *keyrec* data are checked to ensure that they are valid. The following conditions are checked: existence of the zone file, existence of the KSK file, existence of the KSK and ZSK directories, the end-time is greater than one day, and the seconds-count and date string match.

- invalid key data values

  A key's *keyrec* data are checked to ensure that they are valid. The following conditions are checked: valid encryption algorithm, key length falls within algorithm's size range, random generator file exists, and the seconds-count and date string match.

Recognized potential problems include:

- imminent zone expiration

  A zone will expire within one week.

- odd zone-signing date

  A zone's recorded signing date is later than the current system clock.

- orphaned keys

  A key *keyrec* is unreferenced by any set *keyrec*.

- missing key directories

  A zone *keyrec*'s key directories (*kskdirectory* or *zskdirectory*) does not exist.

Recognized inconsistencies include:

- key-specific fields in a zone *keyrec*

  A zone *keyrec* contains key-specific entries. To allow for site-specific extensibility, **krfcheck** does not check for undefined *keyrec* fields.

- zone-specific fields in a key *keyrec*

  A key *keyrec* contains zone-specific entries. To allow for site-specific extensibility, **krfcheck** does not check for undefined *keyrec* fields.

- mismatched zone timestamp

  A zone's seconds-count timestamp does not match its textual timestamp.

- mismatched set timestamp

  A set's seconds-count timestamp does not match its textual timestamp.

- mismatched key timestamp

  A key's seconds-count timestamp does not match its textual timestamp.

OPTIONS

-zone

Only perform checks of zone *keyrec*s. This option may not be combined with the *-set* or *-key* options.

-set

Only perform checks of set *keyrec*s. This option may not be combined with the *-zone* or *-key* options.

-key

Only perform checks of key *keyrec*s. This option may not be combined with the *-set* or *-zone* options.

-count

Display a final count of errors.

-quiet

Do not display messages. This option supersedes the setting of the *-verbose* option.

-verbose

Display many messages. This option is subordinate to the *-quiet* option.

-Version

Display the **krfcheck** version number and exit.

-help

Display a usage message.

SEE ALSO

cleankrf(8), fixkrf(8), lskrf(1), zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3)

file-keyrec(5)

2.3.4 lskrf

NAME

lskrf - List the *keyrec*s in a DNSSEC-Tools *keyrec* file

SYNOPSIS

```
  lskrf [options] <keyrec-files>
```

DESCRIPTION

lskrf lists the contents of the specified *keyrec* files. All *keyrec* files are loaded before the output is displayed. If any *keyrec*s have duplicated names, whether within one file or across multiple files, the later *keyrec* will be the one whose data are displayed.

lskrf has three base output formats. In ascending levels of detail, these formats are terse output, default format, and long format. Terse output is given when the *-terse* option is specified; long output is given when the *-long* option is specified.

The output displayed for each record in a *keyrec* file depends on the selected records, the selected attributes, and the selected output format. Each option in these option groups is described in detail in the OPTIONS section; the three basic output formats are described in the OUTPUT FORMATS section.

OUTPUT FORMATS

keyrec files hold three types of *keyrec* records: zone records, signing set records, and key records. Each type of *keyrec* record contains *keyrec* fields related to that type. For example, zone *keyrec* records contain data about all the keys associated with a particular zone; key *keyrec* records contain key lengths and algorithms for each particular key. The data to be printed must be specified by selecting some combination of the *-zone*, *-set*, *-key*, and *-all* options. There are also options for specifying specific types of keys to be printed.

The three base output formats are the default format, the terse format, and the long format. The *-terse* option indicates that a minimal amount of output is desired; the *-long* option indicates that a great deal of output is desired. The record-selection and attribute-selection options may be used in conjunction with *-terse* to display exactly the set of *keyrec* fields needed. The default output format is a middle ground between terse and long output and is that used when neither *-terse* nor <-long> is given.

Zone *keyrec* Output

The table below shows the zone *keyrec* fields displayed for each output format.

Keyrec Field	**Default**	**Terse**	**Long**
keyrec type	yes	no	yes
zone name	yes	yes	yes
zone file	yes	no	yes
signed zonefile	yes	no	yes
signing date	yes	no	yes
expiration date	no	no	yes
archive directory	no	no	yes
KSK count	no	no	yes
KSK directory	no	no	yes
current KSK set	no	no	yes
published KSK set	no	no	yes
ZSK count	no	no	yes
ZSK directory	no	no	yes
current ZSK set	no	no	yes
published ZSK set	no	no	yes
new ZSK set	no	no	yes

Table 3: **lskrf** Zone Output Formats

Set *keyrec* Output

The table below shows the signing set *keyrec* fields displayed for each output format.

Keyrec Field	**Default**	**Terse**	**Long**
keyrec type	no	no	yes
set name	yes	yes	yes
zone name	yes	no	yes
keys	yes	no	yes
last modification date	no	no	yes

Table 4: **lskrf** Signing Set Output Formats

Key *keyrec* Output

The table below shows the key *keyrec* fields displayed for each output format.

Keyrec Field	**Default**	**Terse**	**Long**
keyrec type	yes	no	yes
key name	yes	yes	yes
algorithm	no	no	yes
end date	no	no	yes
generation date	yes	no	yes
key length	no	no	yes
key life	no	no	yes
key path	no	no	yes
keys	no	no	yes
random number generator	no	no	yes
zone name	yes	no	yes

Table 5: **lskrf** Key Output Formats

OPTIONS

lskrf takes three types of options: record-selection options, record-attribute options, and output-style options. These option sets are detailed below.

Record-selection options are required options; at least one record-selection option **must** be selected. Record-attribute options and output-style options are optional options; any number of these option *may* be selected.

Record-Selection Options

These options select the types of *keyrec* that will be displayed.

-all

This option displays all the records in a *keyrec* file.

-zones

This option displays the zones in a *keyrec* file.

-sets

This option displays the signing sets in a *keyrec* file.

-keys

This option displays the keys in a *keyrec* file.

The key data are sorted by key type in the following order: Current KSKs, Published KSKs, Current ZSKs, Published ZSKs, New ZSKs, Obsolete KSKs, and Obsolete ZSKs.

-ksk

This option displays the KSK keys in a *keyrec* file.

-kcur

This option displays the Current KSK keys in a *keyrec* file.

-kpub

This option displays the Published KSK keys in a *keyrec* file.

-kobs

This option displays the obsolete KSK keys in a *keyrec* file. This option must be give if obsolete KSK keys are to be displayed.

-zsk

This option displays the ZSK keys in a *keyrec* file. It does not include obsolete ZSK keys; the *-obs* option must be specified to display obsolete keys.

-cur

This option displays the Current ZSK keys in a *keyrec* file.

-new

This option displays the New ZSK keys in a *keyrec* file.

-pub

This option displays the Published ZSK keys in a *keyrec* file.

-zobs

This option displays the obsolete ZSK keys in a *keyrec* file. This option must be give if obsolete ZSK keys are to be displayed.

-obs

This option displays the obsolete KSK and ZSK keys in a *keyrec* file. This option is a shorthand method specifying the *-kobs* and *-zobs* options.

Record-Attribute Options

These options select subsets of the *keyrec*s chosen by the record-selection options.

-valid

This option displays the valid zones in a *keyrec* file. It implies the *-zones* option.

-expired

This option displays the expired zones in a *keyrec* file. It implies the *-zones* option.

-ref

This option displays the referenced signing set *keyrec*s and the referenced key *keyrec*s in a *keyrec* file, depending upon other selected options.

Referenced state depends on the following:

- Signing sets are considered to be referenced if they are listed in a zone keyrec.
- KSKs are considered to be referenced if they are listed in a signing set keyrec that is listed in a zone keyrec.
- ZSKs are considered to be referenced if they are listed in a signing set keyrec that is listed in a zone keyrec.

This option may be used with either the *-sets* or *-keys* options. If it isn't used with any record-selection options, then it is assumed that both *-sets* and *-keys* have been specified.

-unref

This option displays the unreferenced signing set *keyrec*s or the unreferenced key *keyrec*s in a *keyrec* file, depending upon other selected options.

Unreferenced state depends on the following:

- Signing sets are considered to be unreferenced if they are not listed in a zone keyrec.
- KSKs are considered to be unreferenced if they are not listed in a signing set keyrec that is listed in a zone keyrec.
- ZSKs are considered to be unreferenced if they are not listed in a signing set keyrec that is listed in a zone keyrec.
- Obsolete ZSKs are checked, whether or not the -obs flag was specified.

This option may be used with either the *-sets* or *-keys* options. If it isn't used with any record-selection options, then it is assumed that both *-sets* and *-keys* have been specified.

Zone-Attribute Options

These options allow specific zone fields to be included in the output. If combined with the *-terse* option, only those fields specifically desired will be printed. These options must be used with the *-zone* option.

-z-archdir

Display the zone's archive directory. If an archive directory is not explicitly set for the zone, the default directory will be listed.

-z-dates

Display the zone's time-stamps. These are the signing date and the expiration date.

-z-dirs

Display the zone's directories. These directories are the KSK directory, the ZSK directory, and the key archive directory.

-z-expdate

Display the zone's expiration date.

-z-ksk

Display the zone's KSK data. This is the equivalent of specifying the *-z-kskcount*, *-z-kskcur*, *-z-kskdir*, and *-z-kskpub* options.

-z-kskcount

Display the zone's KSK count.

-z-kskcur

Display the zone's Current KSK signing set. If this is not defined, then "<unset>" will be given.

-z-kskdir

Display the zone's KSK directory. If this is not defined, then "." will be given.

-z-kskpub

Display the zone's Published KSK signing set. If this is not defined, then "<unset>" will be given.

-z-sets

Display the zone's signing sets. This is the equivalent of specifying the *-z-kskcur*, *-z-kskpub*, *-z-zskcur*, *-z-zsknew*, and *-z-zskpub* options.

-z-signdate

Display the zone's signing date.

-z-signfile

Display the zone's signed zonefile.

-z-zonefile

Display the zone's zonefile.

-z-zsk

Display the zone's ZSK data. This is the equivalent of specifying the *-z-zskcount*, *-z-zskcur*, *-z-zskdir*, *-z-zsknew*, and *-z-zskpub* options.

-z-zskcount

Display the zone's ZSK count.

-z-zskcur

Display the zone's Current ZSK signing set. If this is not defined, then "<unset>" will be given.

-z-zskdir

Display the zone's ZSK directory. If this is not defined, then "." will be given.

-z-zsknew

Display the zone's New ZSK signing set. If this is not defined, then "<unset>" will be given.

-z-zskpub

Display the zone's Published ZSK signing set. If this is not defined, then "<unset>" will be given.

Set-Attribute Options

These options allow specific set fields to be included in the output. If combined with the *-terse* option, only those fields specifically desired will be printed. These options must be used with the *-set* option.

-s-keys

Display the set's keys.

-s-lastmod

Display the set's date of last modification.

-s-zone

Display the set's zone name.

Key-Attribute Options

These options allow specific key fields to be included in the output. If combined with the *-terse* option, only those fields specifically desired will be printed. These options must be used with the *-key* option.

-k-algorithm

Display the key's encryption algorithm.

-k-enddate

Display the key's end-date, calculated by adding the key's lifespan to its signing date.

-k-length

Display the key's length.

-k-lifespan

Display the key's lifespan (in seconds.) This lifespan is **only** related to the time between key roll-over. There is no other lifespan associated with a key.

-k-path

Display the key's path.

-k-random

Display the key's random number generator.

-k-signdate

Display the key's signing date.

-k-zone

Display the key's zonefile.

Output-Format Options

These options define how the *keyrec* information will be displayed.

Without any of these options, the zone name, zone file, zone-signing date, and a label will be displayed for zones. For types, the key name, the key's zone, the key's generation date, and a label will be displayed if these options aren't given.

-count

The count of matching records will be displayed, but the matching records will not be.

-nodate

The key's generation date will not be printed if this flag is given.

-headers

Display explanatory column headers. If this flag is given, then entry labels will not be printed unless explicitly requested by use of the *-label* option.

-label

A label for the *keyrec*'s type will be given.

-long

The long form of output will be given. See the OUTPUT FORMATS section for details on data printed for each type of *keyrec* record.

Long zone output can get *very* wide, depending on the data.

-terse

This options displays only the name of the zones or keys selected by other options.

-help

Display a usage message and exit.

-h-zones

Display the zone-attribute options and exit.

-h-sets

Display the set-attribute options and exit.

-h-keys

Display the key-attribute options and exit.

SEE ALSO

zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3)

file-keyrec(5)

2.3.5 expchk

NAME

expchk - Check a *keyrec* file for expired zones

SYNOPSIS

```
  expchk -all -expired -valid -warn numdays -zone zonename
                -count -help keyrec_files
```

DESCRIPTION

expchk checks a set of *keyrec* files to determine if the zone *keyrec*s are valid or expired. The type of zones displayed depends on the options chosen; if no options are given the expired zones will be listed.

OPTIONS

-all

Display expiration information on all zones, expired or valid, in the specified *keyrec* files.

-expired

Display expiration information on the expired zones in the specified *keyrec* files. This is the default action.

-valid

Display expiration information on the valid zones in the specified *keyrec* files.

-warn numdays

A warning will be given for each valid zone that will expire in *numdays* days. This option has no effect on expired zones.

-zone zonename

Display expiration information on the zone specified in *zonename*.

-count

Only the count of matching zones (valid or expired) will be given. If both types of zones are selected, then the count will be the number of zones in the specified *keyrec* files.

-help

Display a usage message.

SEE ALSO

zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3)

2.3.6 fixkrf

NAME

fixkrf - Fixes DNSSEC-Tools *keyrec* files whose encryption key files have been moved.

SYNOPSIS

```
  fixkrf [options] <keyrec-file> <dir 1> ... <dir N>
```

DESCRIPTION

fixkrf checks a specified *keyrec* file to ensure that the referenced encryption key files exist where listed. If a key is not where the *keyrec* specifies it should be, then **fixkrf** will search the given directories for those keys and adjust the *keyrec* to match reality. If a key of a particular filename is found in multiple places, a warning will be printed and the *keyrec* file will not be changed for that key.

OPTIONS

-list

Display output about missing keys, but don't fix the *keyrec* file.

-verbose

Display output about found keys as well as missing keys.

-help

Display a usage message.

SEE ALSO

cleankrf(8), genkrf(8), lskrf(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3)

file-keyrec.pm(5)

2.3.7 cleanarch

NAME

cleanarch - Clean a DNSSEC-Tools key archive of old keys

SYNOPSIS

```
  cleanarch [options] <keyrec-file | rollrec-file>
```

DESCRIPTION

cleanarch deletes old keys from a DNSSEC-Tools key archive. Key "age" and archives are determined by options and arguments.

Command line options and arguments allow selection of archives, keys to delete, amount of output to provide. The options are divided into three groups: archive selection, key selection, and output format. Complete information on options is provided in the OPTIONS section.

cleanarch takes a single argument (as distinguished from an option.) This argument may be either a *keyrec* file or a *rollrec* file. If the file is a *keyrec* file, the archive directory for its zone *keyrec*s are added to the list of archives to clean. If the file is a *rollrec* file, *keyrec* files for its zones are searched for zone's the archive directory, and those directories are added to the list of archives to clean. If a zone does not have an archive directory explicitly defined, then the DNSSEC-Tools default will be cleaned. The archives specified by this argument may be modified by archive-selection options.

The archive-selection options combine with the *keyrec* or *rollrec* file to select a set of archive directories to clean. (Some options can take the place of the file argument.)

The key-selection options allow the set of keys to be deleted to contain an entire archive, a particular zone's keys, or all the keys prior to a certain date.

The output-format options sets how much output will be given. Without any options selected, the names of keys will be printed as they are deleted. If the *-verbose* option is given, then the directories selected for searching and the keys selected for deletion will be printed. If the *-dirlist* option is given, then the directories selected for searching will be printed and no other action will be taken. If the *-list* option is given, then the keys selected for deletion will be printed and no other action will be taken.

OPTIONS

Archive-Selection Options

The following options allow the user to select the archives to be cleaned.

-archive directory

This option specifies an archive directory to be cleaned.

-defarch

This option indicates that the default archive directory (named in the DNSSEC-Tools configuration file) should be cleaned.

-zone zone

This option indicates that *zone* is the only zone whose archive will be cleaned. If the archive directory is shared by other zones then their keys may also be deleted.

Key-Selection Options

The following options allow the user to select the keys to be deleted.

-all

Deletes all keys in the selected archives. This option may not be used with any other key-selection options.

-days days

Deletes all keys except those whose modification date is within the *days* full days preceding the current day.

-onezone zone

Only keys with *zone* in the key's filename are deleted. This is intended for use in cleaning a multi-zone key archive.

This does not validate that *zone* is an actual zone. **Any** string can be used here. For example, using "private" will select old private key files for deletion and using "com" will select any filename that contains "com".

Options for Output Control

The following options allow the user to control **cleanarch**'s output.

-dirlist

This option lists the selected archive directories. No other action is taken.

-list

This option lists the selected keys. No other action is taken.

-quiet

Display no output.

-verbose

Display verbose output.

-Version

Display the **cleanarch** and DNSSEC-Tools versions.

-help

Display a usage message and exit.

WARNINGS

The user is advised to invest a bit of time testing this tool **prior** to putting it into production use. Once a key is deleted, it is **gone**. Some may find this to be detrimental to the health of their DNSSEC-Tools installation.

SEE ALSO

cleankrf(8), lskrf(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3), Net::DNS::SEC::Tools::rollrec.pm(3)

dnssec-tools.conf(5), keyrec(5), rollrec(5)

2.3.8 cleankrf

NAME

cleankrf - Clean a DNSSEC-Tools *keyrec* files of old data

SYNOPSIS

```
  cleankrf [options] <keyrec-files>
```

DESCRIPTION

cleankrf cleans old data out of a set of DNSSEC-Tools *keyrec* files. The old data are orphaned signing sets, orphaned keys, and obsolete keys.

Orphaned signing sets are set *keyrec*s unreferenced by a zone *keyrec*.

Orphaned keys are KSK key *keyrec*s unreferenced by a zone *keyrec* and ZSK key *keyrec*s unreferenced by any set *keyrec*s.

Obsolete keys are ZSK key *keyrec*s with a *keyrec_type* of **zskobs**.

cleankrf's exit code is the count of orphaned and obsolete *keyrec*s found.

OPTIONS

-count

Display a final count of old *keyrec*s found in the *keyrec* files. This option allows the count to be displayed even if the *-quiet* option is given.

-list

The key *keyrec*s are checked for old *keyrec*s, but they are not removed from the *keyrec* file. The names of the old *keyrec*s are displayed.

-rm

Delete the key files, both **.key** and **.private**, from orphaned and expired *keyrec*s.

-quiet

Display no output.

-verbose

Display output about referenced keys and unreferenced keys.

-help

Display a usage message.

SEE ALSO

fixkrf(8), lskrf(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3)

file-keyrec.pm(5)

2.3.9 signset-editor

NAME

signset-editor - DNSSEC-Tools Signing Set GUI Editor

SYNOPSIS

```
  signset-editor <keyrec-file>
```

DESCRIPTION

signset-editor provides the capability for easy management of signing sets in a GUI. A signing set contains zero or more names of key *keyrec*s. These sets are used by other DNSSEC-Tools utilities for signing zones. The signing sets found in the given *keyrec* file are displayed in a new window. New signing sets may be created and existing signing sets may be modified or deleted from **signset-editor**.

signset-editor has two display modes. The Signing Set Display shows the names of all the set *keyrec*s in the given *keyrec* file. The Keyrec Display shows the names of all the key *keyrec*s in the given *keyrec* file. **signset-editor** starts in Signing Set Display mode, but the mode can be toggled back and forth as needed.

An additional toggle controls the display of additional data. If the Extended Data toggle is turned on, then the Signing Set Display shows the names of the key *keyrec*s in each signing set and the Keyrec Display shows the names of each signing set each key *keyrec* is in. If the Extended Data toggle is turned off, then the Signing Set Display only shows the names of the set *keyrec*s and the Keyrec Display only shows the names key *keyrec*s.

signset-editor has a small number of commands. These commands are all available through the menus, and most have a keyboard accelerator. The commands are described in the next section.

Management of signing sets may be handled using a normal text editor. However, **signset-editor** provides a nice GUI that **only** manipulates signing sets without the potential visual clutter of the rest of the *keyrec* entries.

UNDOING MODIFICATIONS

signset-editor has the ability to reverse modifications it has made to a *keyrec* file. This historical restoration will only work for modifications made during a particular execution of **signset-editor**; modifications made during a previous execution may not be undone.

When undoing modifications, **signset-editor** does not necessarily restore name-ordering within a *keyrec*'s **signing_set** field. However, the signing-set data are maintained. This means that an "undone" *keyrec* file may not be exactly the same, byte-for-byte, as the original file, but the proper meaning of the data is kept.

After a "Save" operation, the data required for reversing modifications are deleted. This is not the case for the "Save As" operation.

COMMANDS

signset-editor provides the following commands, organized by menus:

- **Open** (File menu)

  Open a new *keyrec* file. If the specified file does not exist, the user will be prompted for the action to take. If the user chooses the "Continue" action, then **signset-editor** will continue editing the current *keyrec* file. If the "Quit" action is selected, then **signset-editor** will exit.

- **Save** (File menu)

  Save the current *keyrec* file. The data for the "Undo Changes" command are purged, so this file will appear to be unmodified.

  Nothing will happen if no changes have been made.

- **Save As** (File menu)

  Save the current *keyrec* file to a name selected by the user.

- **Quit** (File menu)

  Exit **signset-editor**.

- **Undo Changes** (Edit menu)

  Reverse modifications made to the signing sets and keyrecs. This is **only** for the in-memory version of the *keyrec* file.

- **New Signing Set** (Commands menu)

  Create a new signing set. The user is given the option of adding key *keyrec*s to the new set.

  This command is available from both viewing modes.

- **Delete Signing Set/Key** (Commands menu)

  Delete the selected signing set or key.

  This command is available from both viewing modes. If used from the Signing Set Display mode, then all the keys in the selected signing set will be removed from that set. If used from the Keyrec Display mode, then the selected key will no longer be part of any signing set.

- **Modify Signing Set/Key** (Commands menu)

  Modify the selected signing set or key.

  This command is available from both viewing modes. If used from the Signing Set Display mode, then the selected signing set may be modified by adding keys to that set or deleting them from that set. If used from the Keyrec Display mode, then the selected key may be added to or deleted from any of the defined signing sets.

- **View Signing Sets** (Display menu)

  The main window will display the *keyrec* file's signing sets. If Extended Data are to be displayed, then each key *keyrec* in the signing set will also be shown. If Extended data are not to be displayed, then only the signing set names will be shown.

This command is a toggle that switches between View Signing Sets mode and View Keyrecs mode.

- **View Keyrecs** (Display menu)

  The main window will display the names of the *keyrec* file's key *keyrec*s. If Extended Data are to be displayed, then the name of each signing set of the *keyrec* will also be shown. If Extended data are not to be shown, then only the *keyrec* names will be displayed.

  This command is a toggle that switches between View Keyrecs mode and View Signing Sets mode.

- **Display Extended Data** (Display menu)

  Additional data will be shown in the main window. For Signing Sets Display mode, the names of the signing set and their constituent key *keyrec*s will be displayed. For Keyrec Display mode, the names of the key *keyrec*s and the Signing Sets it is in will be displayed.

  This command is a toggle that switches between Extended Data display and No Extended Data display.

- **Do Not Display Extended Data** (Display menu)

  No additional data will be shown in the main window. For Signing Sets Display mode, only the names of the Signing Sets will be displayed. For Keyrec Display mode, only the names of the *keyrec*s will be displayed.

  This command is a toggle that switches between No Extended Data display and Extended Data display.

- **Help** (Help menu)

  Display a help window.

KEYBOARD ACCELERATORS

Below are the keyboard accelerators for the **signset-editor** commands:

Accelerator	Function
Ctrl-D	Delete Signing Set
Ctrl-E	Display Extended Data / Do Not Display Extended Data
Ctrl-H	Help
Ctrl-M	Modify Signing Set
Ctrl-N	New Signing Set
Ctrl-O	Open
Ctrl-Q	Quit
Ctrl-S	Save
Ctrl-U	Undo Changes
Ctrl-V	View Signing Sets / View Keyrecs
Ctrl-W	Close Window (New Signing Set, Modify Signing Set, Help)

Table 6: Keyboard Accelerators for **signset-editor**

These accelerators are all lowercase letters.

REQUIREMENTS

signset-editor is implemented in Perl/Tk, so both Perl and Perl/Tk must be installed on your system.

SEE ALSO

cleankrf(8), fixkrf(8), genkrf(8), krfcheck(8), lskrf(1), zonesigner(8)

Net::DNS::SEC::Tools::keyrec(3)

file-keyrec(5)

2.4 Zone-Rollover Commands

The zone-rollover commands provide tools to assist in DNSSEC zone rollover and keeping records about those zones' rollover status. The commands in this group with together with the Zone-Signing Commands (see Section 2.3.) These commands are:

rollerd	the DNSSEC-Tools key rollover daemon
rollinit	create a new *rollrec* file
rollchk	verify the validity of the contents of a *rollrec* file
rollset	modify entries in a *rollrec* file
rollrec-editor	GUI editor for *rollrec* files
lsroll	list the contents of a *rollrec* file
rollctl	communicate with the **rollerd** rollover daemon
rolllog	write a message to the rollover log file
blinkenlights	GUI tool for monitoring and controlling **rollerd**

2.4.1 rollerd

NAME

rollerd - DNSSEC-Tools daemon to manage DNSSEC key rollover

SYNOPSIS

```
  rollerd [-options] -rrfile <rollrec_file>
```

DESCRIPTION

The **rollerd** daemon manages key rollover for zones. **rollerd** handles both KSK and ZSK rollover, though only one rollover may take place at a time. Initiation of KSK rollovers takes precedence over the initiation of ZSK rollovers. The Pre-Publish Method of key rollover is used for ZSK key rollovers. The Double Signature Method of key rollover is used for KSK rollovers. **rollerd** maintains zone rollover state in files called *rollrec* files. The administrator may control **rollerd** with the **rollctl** command. These are described in their own sections below.

ZSK Rollover Using the Pre-Publish Method

The Pre-Publish Method has four phases that are entered when it is time to perform ZSK rollover:

1. wait for old zone data to expire from caches
2. sign the zone with the KSK and Published ZSK
3. wait for old zone data to expire from caches
4. adjust keys in keyrec and sign the zone with new Current ZSK

rollerd uses the **zonesigner** command during ZSK rollover phases 2 and 4. **zonesigner** will generate keys as required and sign the zone during these two phases.

The Pre-Publish Method of key rollover is defined in the Step-by-Step DNS Security Operator Guidance Document. See that document for more detailed information.

KSK Rollover Using the Double Signature Method

The Double Signature Method has seven phases that are entered when it is time to perform ZSK rollover:

1. wait for old zone data to expire from caches
2. generate a new (published) KSK
3. wait for the old DNSKEY RRset to expire from caches
4. roll the KSKs

5. transfer new keyset to the parent
6. wait for parent to publish the new DS record
7. reload the zone

rollerd uses the **zonesigner** command during KSK rollover phases 2 and 4. **zonesigner** will generate keys as required and sign the zone during these two phases.

Currently, steps 5 and 6 are handled manually. In step 5, **rollerd** informs the administrator that the zone's keyset must be transferred to its parent in order for rollover to continue. In step 6, after the parent has published a new DS record, the administrator uses **rollctl** to inform **rollerd** that the DS record has been published and rollover may continue.

The Double Signature Method of key rollover is defined in the Step-by-Step DNS Security Operator Guidance Document. See that document for more detailed information.

rollrec Files

The zones to be managed by **rollerd** are described in a *rollrec* file. Each zone's entry contains data needed by **rollerd** and some data useful to a user. Below is a sample *rollrec* entry:

```
roll "example.com"
        zonefile        "example.com.signed"
        keyrec          "example.com.krf"
        directory        "dir-example.com"
        kskphase        "0"
        zskphase        "3"
        ksk_rollsecs    "1172614842"
        ksk_rolldate    "Tue Feb 27 22:20:42 2007"
        zsk_rollsecs    "1172615087"
        zsk_rolldate    "Tue Feb 27 22:24:47 2007"
        maxttl          "60"
        display         "1"
        phasestart      "Tue Feb 27 22:25:07 2007"
```

The first line gives the *rollrec* entry's name. The following lines give the zone's signed zone file, *keyrec* file, the current rollover phases, the rollover timestamps, and other information.

If either of the *zonefile* or *keyrec* files do not exist, then a "roll" *rollrec* will be changed into a "skip" *rollrec*. That record will not be processed.

A more detailed explanation may be found in *rollrec(5)*.

Directories

rollerd's execution directory is either the directory in which it is executed or the directory passed in the *-directory* command-line option. Any files used by **rollerd** that were not specified with absolute paths use this directory as their base.

The *directory* field informs **rollerd** where the zone's files may be found. For that zone, **rollerd** will move into that directory, then return to its execution directory when it finishes rollover operations for that zone. If the *directory* value is a relative path, it will be appended to **rollerd**'s execution directory. If the *directory* value is an absolute path, it will be used as is.

Controlling rollerd with rollctl

The **rollctl** command is used to control the behavior of **rollerd**. A number of commands are available, such as starting or stopping rollover for a selected zone or all zones, turning on or off a GUI rollover display, and halting **rollerd** execution. The communications path between **rollerd** and **rollctl** is operating system-dependent. On Unix-like systems, it is a Unix pipe that should **only** be writable by root. A more detailed explanation of **rollctl** may be found in **rollctl(8)**.

A Note About Files and Filenames

There are a number of files and filenames used by **rollerd** and **zonesigner**. The user must be aware of the files used by these programs, where the files are located, and where the programs are executed.

By default, **rollerd** will change directory to the DNSSEC-Tools directory, though this may be changed by the *-directory* option. Any programs started by **rollerd**, most importantly **zonesigner**, will run in this same directory. If files and directories referenced by these programs are named with relative paths, those paths must be relative to this directory.

The *rollrec* entry name is used as a key to the *rollrec* file and to the zone's *keyrec* file. This entry does not have to be the name of the entry's domain, but it is a very good idea to make it so. Whatever is used for this entry name, the same name **must** be used for the zone *keyrec* in that zone's *keyrec* file.

It is probably easiest to store *rollrec* files, *keyrec* files, zone files, and key files in a single directory.

INITIALIZATION AND USAGE

The following steps must be taken to initialize and use **rollerd**. This assumes that zone files have been created, and that BIND and DNSSEC-Tools have been installed.

0. sign zones

   The zones to be managed by **rollerd** must be signed. Use **zonesigner** to create the signed zone files and the *keyrec* files needed by **rollerd**. The *rollrec* file created in the next step **must** use the *keyrec* file names and the signed zone file names created here.

1. create *rollrec* file

   Before **rollerd** may be used, a *rollrec* file must first be created. While this file may be built by hand, the **rollinit** command was written specifically to build the file.

2. select operational parameters

   A number of **rollerd**'s operational parameters are taken from the DNSSEC-Tools configuration file. However, these may be overridden by command-line options. See the OPTIONS section below for more details. If non-standard parameters are desired to always be used, the appropriate fields in the DNSSEC-Tools configuration file may be modified to use these values.

3. install the rollover configuration

   The complete rollover configuration – **rollerd**, *rollrec* file, DNSSEC-Tools configuration file values, zone files – should be installed. The appropriate places for these locations are both installation-dependent and operating system-dependent.

4. test the rollover configuration

   The complete rollover configuration should be tested.

   Edit the zone files so that their zones have short TTL values. A minute TTL should be sufficient. Test rollovers of this speed should **only** be done in a test environment without the real signed zone.

   Run the following command:

   ```
   rollerd -rrfile test.rollrec -logfile - -loglevel info -sleep 60
   ```

   This command assumes the test *rollrec* file is **test.rollrec**. It writes a fair amount of log messages to the terminal, and checks its queue every 60 seconds. Follow the messages to ensure that the appropriate actions, as required by the Pre-Publish Method, are taking place.

5. set **rollerd** to start at boot

   Once the configuration is found to work, **rollerd** should be set to start at system boot. The actual operations required for this step are operating system-dependent.

6. reboot and verify

   The system should be rebooted and the **rollerd** logfile checked to ensure that **rollerd** is operating properly.

OPTIONS

The following options are recognized:

-rrfile rollrec_file

Name of the *rollrec* file to be processed. This is the only required "option".

-directory dir

Sets the **rollerd** execution directory. This must be a valid directory.

-logfile log_file

Sets the **rollerd** log file to *log_file*. This must be a valid logging file, meaning that if *log_file* already exists, it must be a regular file. The only exceptions to this are if *logfile* is **/dev/stdout**, **/dev/tty**, **-**. Of these three, using a *log_file* of **-** is preferable since Perl will properly convert the **-** to the process' standard output.

-loglevel level

Sets **rollerd**'s logging level to *level.* **rollmgr.pm(3)** contains a list of the valid logging levels.

-sleep sleeptime

Sets **rollerd**'s sleep time to *sleeptime.* The sleep time is the amount of time (in seconds) **rollerd** waits between processing its *rollrec*-based queue.

-parameters

Prints a set of **rollerd** parameters and then exits.

-display

Starts the **blinkenlights** graphical display program to show the status of zones managed by **rollerd**.

-help

Display a usage message.

-verbose

Verbose output will be given.

ASSUMPTIONS

rollerd uses the **rndc** command to communicate with the BIND **named** daemon. Therefore, it assumes that appropriate measures have been taken so that this communication is possible.

KNOWN PROBLEMS

The following problems (or potential problems) are known:

- Any process that can write to the rollover socket can send commands to **rollerd**. This is probably not a Good Thing.
- Very little testing was done with zone files and key files not in the process' directory.

POSSIBLE ENHANCEMENTS

The following potential enhancements may be made:

- It'd be good to base **rollerd**'s sleep time on when the next operation must take place, rather than a simple seconds count.

SEE ALSO

blinkenlights(8), named(8), rndc(8), rollchk(8), rollctl(8), rollinit(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3), Net::DNS::SEC::Tools::defaults.pm(3), Net::DNS::SEC::Tools::keyrec.pm(3), Net::DNS::SEC::Tools::rolllog.pm(3), Net::DNS::SEC::Tools::rollmgr.pm(3), Net::DNS::SEC::Tools::rollrec.pm(3)

rollrec(5)

2.4.2 rollinit

NAME

rollinit - Create new *rollrec* records for a DNSSEC-Tools *rollrec* file

SYNOPSIS

```
  rollinit [options] <zonename1> ... <zonenameN>
```

DESCRIPTION

rollinit creates new *rollrec* entries for a *rollrec* file. This *rollrec* file will be used by **rollerd** to manage key rollover for the named domains.

A *rollrec* entry has this format:

```
    roll "example.com"
        zonefile        "example.com.signed"
        keyrec          "example.com.krf"
        kskphase        "0"
        zskphase        "0"
        administrator   "bob@bobhost.example.com"
        directory       "/var/dns/zones/example.com"
        loglevel        "phase"
        ksk_rolldate    " "
        ksk_rollsecs    "0"
        zsk_rolldate    " "
        zsk_rollsecs    "0"
        maxttl          "604800"
        display         "1"
        phasestart      "Mon Jan 9 16:00:00 2006"
```

The *zonefile* and *keyrec* fields are set according to command-line options and arguments. The manner of generating the *rollrec*'s actual values is a little complex and is described in the ZONEFILE And KEYREC FIELDS section below.

The *administrator* field is set to "bobbobhost.example.com" to indicate that the email messages to the zone's administrator should be sent to "bobbobhost.example.com".

The *directory* field is set to "/var/dns/zones/example.com" to indicate that the files for this zone should be found in **/var/dns/zones/example.com**. This includes the zone file, the signed zone file, and the *keyrec* file.

The *loglevel* field is set to "phase" to indicate that **rollerd** should only log phase-level (and greater) log messages for this zone.

The *kskphase* field is set to 0 to indicate that the zone is in normal operation (non-rollover) for KSK keys. The *zskphase* field is set to 0 to indicate that the zone is in normal operation (non-rollover) for ZSK keys.

The *ksk_rolldate* and *ksk_rollsecs* fields are set to indicate that the zone has not yet undergone KSK rollover.

The *zsk_rolldate* and *zsk_rollsecs* fields are set to indicate that the zone has not yet undergone ZSK rollover.

The *display* field is set to indicate that **blinkenlights** should display the record. The *maxttl* and *phasestart* fields are set to dummy values.

The keywords **roll** and **skip** indicate whether **rollerd** should process or ignore a particular *rollrec* entry. **roll** records are created by default; **skip** entries are created if the *-skip* option is specified.

The newly generated *rollrec* entries are written to standard output, unless the *-out* option is specified.

ZONEFILE and KEYREC FIELDS

The *zonefile* and *keyrec* fields may be given by using the *-zone* and *-keyrec* options, or default values may be used.

The default values use the *rollrec*'s zone name, taken from the command line, as a base. **.signed** is appended to the domain name for the zone file; **.krf** is appended to the domain name for the *keyrec* file.

If *-zone* or *-keyrec* are specified, then the options values are used in one of two ways:

- A single domain name is given on the command line.

  The option values for *-zone* and/or *-keyrec* are used for the actual *rollrec* fields.

- Multiple domain names are given on the command line.

  The option values for *-zone* and/or *-keyrec* are used as templates for the actual *rollrec* fields. The option values must contain the string =. This string is replaced by the domain whose *rollrec* is being created.

See the EXAMPLES section for examples of how options are used by **rollinit**.

OPTIONS

rollinit may be given the following options:

-zone zonefile

This specifies the value of the *zonefile* field. See the ZONEFILE And KEYREC FIELDS and EXAMPLES sections for more details.

-keyrec keyrec-file

This specifies the value of the *keyrec* field. See the ZONEFILE And KEYREC FIELDS and EXAMPLES sections for more details.

-admin

This specifies the value of the *administrator* field. If it is not given, an *administrator* field will not be included for the record.

-directory

This specifies the value of the *directory* field. If it is not given, a *directory* field will not be included for the record.

-loglevel

This specifies the value of the *loglevel* field. If it is not given, a *loglevel* field will not be included for the record.

-skip

By default, **roll** records are generated. If this option is given, then **skip** records will be generated instead.

-out output-file

The new *rollrec* entries will be appended to *output-file*. The file will be created if it does not exist.

If this option is not given, the new *rollrec* entries will be written to standard output.

-help

Display a usage message.

-Version

Display version information for **rollinit** and DNSSEC-Tools.

EXAMPLES

The following options should make clear how **rollinit** deals with options and the new *rollrec*s. Example 1 will show the complete new *rollrec* record. For the sake of brevity, the remaining examples will only show the newly created *zonefile* and *keyrec* records.

Example 1. One domain, no options

This example shows the *rollrec* generated by giving **rollinit** a single domain, without any options.

```
$ rollinit example.com
    roll    "example.com"
        zonefile        "example.com.signed"
        keyrec          "example.com.krf"
        kskphase        "0"
        zskphase        "0"
        ksk_rolldate    " "
        ksk_rollsecs    "0"
        zsk_rolldate    " "
        zsk_rollsecs    "0"
        maxttl          "0"
        display         "1"
        phasestart      "new"
```

Example 2. One domain, -zone option

This example shows the *rollrec* generated by giving **rollinit** a single domain, with the *-zone* option.

```
    $ rollinit -zone signed-example example.com
        roll    "example.com"
            zonefile        "signed-example"
            keyrec          "example.com.krf"
```

Example 3. One domain, -keyrec option

This example shows the *rollrec* generated by giving **rollinit** a single domain, with the *-keyrec* option.

```
    $ rollinit -keyrec x-rrf example.com
        roll    "example.com"
            zonefile        "example.com.signed"
            keyrec          "x-rrf"
```

Example 4. One domain, -zone and -keyrec options

This example shows the *rollrec* generated by giving **rollinit** a single domain, with the *-zone* and *-keyrec* options.

```
    $ rollinit -zone signed-example -keyrec example.rrf example.com
        roll    "example.com"
            zonefile        "signed-example"
            keyrec          "example.rrf"
```

Example 5. One domain, -skip option

This example shows the *rollrec* generated by giving **rollinit** a single domain, with the *-zone* and *-keyrec* options.

```
    $ rollinit -skip example.com
        skip    "example.com"
            zonefile        "example.com.signed"
            keyrec          "example.com.krf"
```

Example 6. Multiple domains, no options

This example shows the *rollrec*s generated by giving **rollinit** several domains, without any options.

```
$ rollinit example1.com example2.com
    roll    "example1.com"
            zonefile        "example1.com.signed"
            keyrec          "example1.com.krf"

    roll    "example2.com"
            zonefile        "example2.com.signed"
            keyrec          "example2.com.krf"
```

Example 7. Multiple domains, -zone option

This example shows the *rollrec*s generated by giving **rollinit** several domains, with the -*zone* option.

```
$ rollinit -zone =-signed example1.com example2.com
    roll    "example1.com"
            zonefile        "example1.com-signed"
            keyrec          "example1.com.krf"

    roll    "example2.com"
            zonefile        "example2.com-signed"
            keyrec          "example2.com.krf"
```

Example 8. Multiple domains, -keyrec option

This example shows the *rollrec*s generated by giving **rollinit** several domains, with the -*keyrec* option.

```
$ rollinit -keyrec zone-=-keyrec example1.com example2.com
    roll    "example1.com"
            zonefile        "example1.com.signed"
            keyrec          "zone-example1.com-keyrec"

    roll    "example2.com"
            zonefile        "example2.com.signed"
            keyrec          "zone-example2.com-keyrec"
```

Example 9. Multiple domains, -zone and -keyrec options

This example shows the *rollrec*s generated by giving **rollinit** several domains, with the -*zone* and -*keyrec* options.

```
$ rollinit -zone Z-= -keyrec =K example1.com example2.com
    roll    "example1.com"
            zonefile        "Z-example1.com"
            keyrec          "example1.comK"
```

```
        roll    "example2.com"
                zonefile        "Z-example2.com"
                keyrec          "example2.comK"
```

Example 10. Single domain, -zone and -keyrec options with template

This example shows the *rollrec* generated by giving **rollinit** a single domain, with the *-zone* and *-keyrec* options. The options use the multi-domain = template.

```
    $ rollinit -zone Z-= -keyrec =.K example.com
        roll    "example.com"
                zonefile        "Z-="
                keyrec          "=.K"
```

This is probably not what is wanted, since it results in the *zonefile* and *keyrec* field values containing the =.

Example 11. Multiple domains, -zone and -keyrec options without template

This example shows the *rollrec*s generated by giving **rollinit** several domains, with the *-zone* and *-keyrec* options. The options do not use the multi-domain = template.

```
    $ rollinit -zone ex.zone -keyrec ex.krf example1.com example2.com
        roll    "example1.com"
                zonefile        "ex.zone"
                keyrec          "ex.krf"

        roll    "example2.com"
                zonefile        "ex.zone"
                keyrec          "ex.krf"
```

This may not be what is wanted, since it results in the same *zonefile* and *keyrec* fields values for each *rollrec.*

SEE ALSO

lsroll(1)

rollerd(8), rollchk(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec.pm(3), Net::DNS::SEC::Tools::rollrec.pm(3)

Net::DNS::SEC::Tools::file-keyrec.pm(5), Net::DNS::SEC::Tools::file-rollrec.pm(5)

2.4.3 rollchk

NAME

rollchk - Check a DNSSEC-Tools *rollrec* file for problems and inconsistencies

SYNOPSIS

```
  rollchk [-roll | -skip] [-count] [-quiet] [-verbose] [-help] rollrec-file
```

DESCRIPTION

This script checks the *rollrec* file specified by *rollrec-file* for problems and inconsistencies.

Recognized problems include:

- non-existent rollrec file

  The specified *rollrec* file does not exist.

- no zones defined

  No zones are defined in the specified *rollrec* file.

- invalid KSK rollover phase

  A zone has an invalid KSK rollover phase. These phases may be 0, 1, 2, 3, 4, 5, 6, or 7; any other value is invalid.

- mismatch in KSK timestamp data

  A zone's KSK roll-seconds timestamp does not translate into the date stored in its roll-date string.

- invalid ZSK rollover phase

  A zone has an invalid ZSK rollover phase. These phases may be 0, 1, 2, 3, or 4; any other value is invalid.

- mismatch in ZSK timestamp data

  A zone's ZSK roll-seconds timestamp does not translate into the date stored in its roll-date string.

- contemporaneous KSK and ZSK rollovers

  A zone has a KSK rollover occurring at the same time as a ZSK rollover. A zone may only have one rollover phase be non-zero at a time.

- in rollover without a phasestart

  A zone is currently in rollover, but its *rollrec* record does not have a *phasestart* field.

- empty administrator

  A zone has an empty administrator field. This field must contain an email address.

- non-existent directory

  Several checks are made for a zone's directory. If the zone has a directory specified, the directory must exist and it must be an actual directory.

- invalid display flag

  A zone has an invalid display flag. This flag may be 0 or 1; any other value is invalid.

- non-positive maxttl

  The maximum TTL value must be greater than zero.

- zone file checks

  Several checks are made for a zone's zone file. The zone file must exist, it must be a regular file, and it must not be of zero length.

- keyrec file checks

  Several checks are made for a zone's *keyrec* file. The *keyrec* file must exist, it must be a regular file, and it must not be of zero length.

OPTIONS

-roll

Only display *rollrec*s that are active ("roll") records. This option is mutually exclusive of the *-skip* option.

-skip

Only display *rollrec*s that are inactive ("skip") records. This option is mutually exclusive of the *-roll* option.

-count

Display a final count of errors.

-quiet

Do not display messages. This option supersedes the setting of the *-verbose* option.

-verbose

Display many messages. This option is subordinate to the *-quiet* option.

-help

Display a usage message.

SEE ALSO

lsroll(8), rollerd(8), rollinit(8)

Net::DNS::SEC::Tools::rollrec.pm(3)

file-rollrec(5)

2.4.4 rollset

NAME

rollset - Modifies entries in a DNSSEC-Tools *rollrec* file

SYNOPSIS

```
  rollset [options] rollrec-file
```

DESCRIPTION

rollset modifies fields in the *rollrec* file specified by *rollrec-file.* Multiple options may be combined in a single **rollset** execution. **rollset** operates quietly unless it is given the -*verbose* option.

All records in the specified *rollrec* file will be modified, unless the -*zone* option is given. In that case, only the named zone will be modified.

OPTIONS

-zone zonename

The zone *zonename* is selected as the only zone whose *rollrec* record will be modified. If this is not given, then all *rollrec* records will be modified.

-file zone-file

The zone file in the selected *rollrec* records is modified to be *zone-file.*

-keyrec keyrec-file

The keyrec file in the selected *rollrec* records is modified to be *keyrec-file.*

-admin addr

The zone administrator's email address is set to *addr.*

-del-admin

The *administrator* line is deleted.

-del-directory

The *directory* line is deleted.

-del-loglevel

The *loglevel* line is deleted.

-directory dir

The directory to hold the zone's files is set to *dir.*

-loglevel logging-level

The logging level of the selected *rollrec* records is set to *logging-level.* The valid logging levels are defined in **rollmgr.pm(3)**.

-display

Turn on the GUI display of the zones in the selected *rollrec*s. This option is mutually exclusive of the *-nodisplay* option.

-nodisplay

Turn off the GUI display of the zones in the selected *rollrec*s. This option is mutually exclusive of the *-display* option.

-roll

Convert the selected *rollrec*s to be active ("roll") records. This option is mutually exclusive of the *-skip* option.

-skip

Convert the selected *rollrec*s to be inactive ("skip") records. This option is mutually exclusive of the *-roll* option.

-check

If this option is given, the **rollchk** command will be run on the modified *rollrec* file.

-verbose

Display information about every modification made to the *rollrec* file.

-Version

Display the current **rollset** and DNSSEC-Tools versions.

-help

Display a usage message.

SEE ALSO

lsroll(8), rollchk(8), rollerd(8), rollinit(8)

Net::DNS::SEC::Tools::rollmgr.pm(3), Net::DNS::SEC::Tools::rollrec.pm(3)

file-rollrec(5)

2.4.5 rollrec-editor

NAME

rollrec-editor - DNSSEC-Tools Rollrec GUI Editor

SYNOPSIS

```
  rollrec-editor [options] <rollrec-file>
```

DESCRIPTION

rollrec-editor provides the capability for easy GUI-based management of *rollrec* files. A *rollrec* file contains one or more *rollrec* records. These records are used by the DNSSEC-Tools rollover utilities (**rollerd**, etc.) to describe zones' rollover state. Each zone's *rollrec* record contains such information as the zone file, the rollover phase, and logging level. *rollrec* files are text files and may be edited by any text editor. **rollrec-editor** allows editing of only those records a user should change and performs error checking on the data.

When **rollrec-editor** starts, a window is created that has "buttons" for each *rollrec* record in the given *rollrec* file. (In this documentation, this window is called the Button Window.) Clicking on the buttons selects (or deselects) that zone. After one or more zones are selected, one of several commands may be executed. Commands allow modification and deletion of existing *rollrec* records, creation of new *rollrec* records, merging of *rollrec* files, and verification of file validity.

rollrec-editor's commands are available through the menus and most have a keyboard accelerator. The commands are described below in the COMMANDS section.

When a *rollrec* record is selected for modification, a new window is created to hold the editable fields of the record. The fields may be modified in place. When editing is complete, the record is "saved". This does not save the modified *rollrec* into its on-disk file; the file must be saved explicitly from the Button Window.

As stated above, verification checks are performed when saving an edited *rollrec* record. One set of checks ensures that files and directories associated with a zone actually exist. This check may be turned off at command start-up with the *-ignore-warns* command line option. It may be modified during execution with the "Ignore Edit Warnings" menu command.

Button Window Layout

The Button Window contains a button for each *rollrec* record in the selected file. The buttons are arranged in a table that with at least three columns. The number of columns may be set at command start-up with the *-columns* command line option. It may be modified during execution with the "Columns in Button Window" menu command.

When setting the number of columns, **rollrec-editor** minimizes the space required to display the selected file's *rollrec* buttons. This will sometimes cause the number of columns to differ from that requested.

For example, assume a *rollrec* file has 12 *rollrec* records. The following table shows how many rows and columns are given for each of the given column selections.

Column Count	**Rows**	**Columns**
column count	rows	columns
1	12	1
2	6	2
3	4	3
4	3	4
5	3	4
6	2	6
7	2	6
8	2	6
9	2	6
10	2	6
11	2	6
12	1	12

Table 7: Example Button Layouts

The actual rows and columns for a requested column count will vary in order to allow a "best-fit".

UNDOING MODIFICATIONS

The "undo" capability is not currently implemented.

rollrec-editor has the ability to reverse modifications it has made to a *rollrec* file. This historical restoration will only work for modifications made during a particular execution of **rollrec-editor**; modifications made during a previous execution may not be undone.

After a "Save" operation, the data required for reversing modifications are deleted. This is not the case for the "Save As" operation.

OPTIONS

rollrec-editor supports the following options.

- **-columns columns**

  This option allows the user to specify the number of columns to be used in the Button Window.

- **-ignore-warns**

  This option causes **rollrec-editor** to ignore edit warnings when performing validation checks on the contents of *rollrec* records.

- **-no-filter**

  This option turns off name filtering when **rollrec-editor** presents a file-selection dialog for choosing a new *rollrec* file. If this option is not given, then the file-selection dialog will only list regular files with a suffix of **.rrf**.

- **-V**

This option displays the program and DNSSEC-Tools version.

COMMANDS

rollrec-editor provides the following commands, organized by menus: File, Edit, Commands, Rollrecs, and Options.

File Menu

The File Menu contains commands for manipulating *rollrec* files and stopping execution.

- **Open**

  Open a new *rollrec* file. If the specified file does not exist, the user will be prompted for the action to take. If the user chooses the "Continue" action, then **rollrec-editor** will continue editing the current *rollrec* file. If the "Quit" action is selected, then **rollrec-editor** will exit.

- **Save**

  Save the current *rollrec* file. The data for the "Undo Changes" command are purged, so this file will appear to be unmodified.

  Nothing will happen if no changes have been made.

- **Save As**

  Save the current *rollrec* file to a name selected by the user.

- **Quit**

  Exit **rollrec-editor**.

Edit Menu

The Edit Menu contains commands for general editing operations.

- **Undo Changes**

  Reverse modifications made to the *rollrec* records. This is **only** for the in-memory version of the *rollrec* file.

  Not currently implemented.

Commands Menu

The Commands Menu contains commands for modifying *rollrec* records.

- **New Rollrec**

  Create a new *rollrec* record. The user is given a new window in which to edit the user-modifiable *rollrec* fields. A button for the new *rollrec* record will be inserted into the Button Window.

After editing is completed, the "Save" button will add the new *rollrec* record to the in-memory *rollrec* file. The file must be saved in order to have the new *rollrec* added to the file.

Potentially erroneous conditions will be reported to the user at save time. If the *ignore-warnings* flag has been turned on, then these warnings will not be reported. Errors (e.g., invalid log conditions) will always be reported.

- **Delete Selected Rollrecs**

  Delete the selected *rollrec* records. The buttons for each selected record will be removed from the Button Window.

- **Edit Selected Rollrecs**

  Modify the selected *rollrec* records. For every record selected for modification, the user is given a new window in which to edit the user-modifiable *rollrec* fields. When the edit window goes away (after a "Save" or "Cancel"), the *rollrec* record's button is deselected.

  After editing is completed, the "Save" button will add the new *rollrec* record to the in-memory *rollrec* file. The file must be saved in order to have the new *rollrec* added to the file.

  Potentially erroneous conditions will be reported to the user at save time. If the *ignore-warnings* flag has been turned on, then these warnings will not be reported. Errors (e.g., invalid log conditions) will always be reported.

- **Merge Rollrec Files**

  Merge a *rollrec* file with the currently open *rollrec* file. After a successful merge, the *rollrec* records in the second file will be added to the *end* of the currently open *rollrec* file.

  If there are any *rollrec* name collisions in the files, then the user will be asked whether to continue with the merge or cancel it. If the merge continues, then the conflicting *rollrec* records from the "new" file will be discarded in favor of the currently open *rollrec* file.

- **Verify Rollrec File**

  Verify the validity of the *rollrec* file. The user-editable fields in the open *rollrec* file are checked for validity. An edit window is opened for each *rollrec* record that registers an error or warning.

  If the *ignore-warnings* flag has been turned on, then potentially erroneous conditions will not be reported. Errors (e.g., invalid log conditions) will always be reported.

- **Summarize Problems**

  Summarize the warning and error counts of the *rollrec* file. Each *rollrec* record in the open *rollrec* file is checked for validity. If any warnings or errors are found, a window is displayed that lists the name of each *rollrec* record and its warning and error counts.

  If the *ignore-warnings* flag has been turned on, then potentially erroneous conditions will not be reported. Errors (e.g., invalid log conditions) will always be reported.

View Menu

The View Menu contains miscellaneous commands for viewing edit windows.

- **Select All Rollrecs/Unselect All Rollrecs**

  Select or unselect all *rollrec* buttons. All the buttons in the Button Window will be selected or unselected. If **any** of the buttons are not selected, this command will cause all the buttons to become selected. If *all* of the buttons are selected, this command will cause all the buttons to be deselected.

  This command is a toggle that switches between Select All Rollrecs and Unselect All Rollrecs.

- **Reveal Rollrec Edit Windows**

  Raise all *rollrec* edit windows. This command brings all *rollrec* edit windows to the front so that any that are hidden behind other windows will become visible.

- **Close Rollrec Edit Windows**

  Close all *rollrec* edit windows. This command closes and deselects all *rollrec* edit windows.

Options Menu

The Options Menu allows the user to set several options that control **rollrec-editor** behavior.

- **Ignore Edit Warnings/Don't Ignore Edit Warnings**

  Certain operations perform validation checks on the contents of *rollrec* records. Warnings and errors may be reported by these checks. This option controls whether or not warnings are flagged during validation. If they are flagged, then the operation will not continue until the warning condition is fixed or the operation canceled. If the warnings are ignored, then the operation will complete without the condition being fixed.

  This command is a toggle that switches between Ignore Edit Warnings mode and Don't Ignore Edit Warnings mode.

- **Filter Name Selection/Don't Filter Name Selection**

  When opening *rollrec* files for editing or merging, a file-selection dialog box is displayed to allow the user to select a *rollrec* file. This option controls whether a filename filter is used by the dialog box. If Filter Names Selection mode is used, then only regular files with a suffix of **.rrf** will be displayed by the dialog box. If Don't Filter Name Selection mode is used, then all regular files will be displayed by the dialog box.

  This command is a toggle that switches between Filter Name Selection display and Don't Filter Name Selection display.

- **Columns in Button Window**

  Set the number of columns used in the Button Window. See the Button Window Layout section for more information on columns in the Button Window.

Help Menu

The Help Menu contains commands for getting assistance.

- **Help**

  Display a help window.

KEYBOARD ACCELERATORS

Below are the keyboard accelerators for the **rollrec-editor** commands:

Accelerator	**Function**
Ctrl-A	Select All Rollrecs
Ctrl-D	Delete Selected Rollrecs
Ctrl-E	Edit Selected Rollrecs
Ctrl-H	Help
Ctrl-K	Close Rollrec Edit Windows
Ctrl-M	Merge Rollrec Files
Ctrl-N	New Rollrec
Ctrl-O	Open
Ctrl-Q	Quit
Ctrl-R	Reveal Rollrec Edit Windows
Ctrl-S	Save
Ctrl-U	Undo Changes
	(not currently implemented)
Ctrl-V	Verify Rollrec File

Table 8: Keyboard Accelerators for **rollrec-editor**

These accelerators are all lowercase letters.

REQUIREMENTS

rollrec-editor is implemented in Perl/Tk, so both Perl and Perl/Tk must be installed on your system.

KNOWN ISSUES

The following are known issues. These will be resolved in the fullness of time.

There is an issue with the column count and adding new *rollrec*s. It doesn't always work properly, thus occasionally leaving some *rollrec* buttons undisplayed.

Segmentation violation on exit. Should not affect anything as file has already been saved. Occurs on MacOS X 10.4.9 with Perl 5.8.6; does not occur on Linux FC5 with Perl 5.8.8.

There is no undo command.

SEE ALSO

lsroll(1)

rollchk(8), rollerd(8) rollinit(8), rollset(8), zonesigner(8)

Net::DNS::SEC::Tools::rollrec(3)

file-rollrec(5)

2.4.6 lsroll

NAME

lsroll - List the *rollrec*s in a DNSSEC-Tools *rollrec* file

SYNOPSIS

```
  lsroll [options] <rollrec-files>
```

DESCRIPTION

This script lists the contents of the specified *rollrec* files. All *rollrec* files are loaded before the output is displayed. If any *rollrec*s have duplicated names, whether within one file or across multiple files, the later *rollrec* will be the one whose data are displayed.

OUTPUT FORMATS

The output displayed for each zone in a *rollrec* file depends on the selected records, the selected attributes, and the selected output format. Each option in these option groups is described in detail in the next section. The three base output formats, along with the default *-skip* format, are described here.

The *-terse* option indicates that a minimal amount of output is desired; the *-long* option indicates that a great deal of output is desired. The record-selection and attribute-selection options may be used in conjunction with *-terse* to display exactly the set of *rollrec* fields needed.

The default output format is that used when neither *-terse* nor *-long* is given, and is a middle ground between terse and long output.

If the *-skip* option is given, then the default output format is a little more restricted than the normal default. Some *rollrec* fields don't make sense in the context of a skip records, and so are given as "—". These fields are the KSK rollover phase, the ZSK rollover phase, the TTL value, and the phase start.

The table below shows the fields displayed for each output format.

Rollrec Field	**Default**	**Terse**	**Long**	**Skip**
rollrec name	yes	yes	yes	yes
rollrec type	no	no	yes	no
zone name	yes	no	yes	yes
keyrec file	yes	no	yes	yes
KSK phase	yes	no	yes	no
ZSK phase	yes	no	yes	no
administrator	no	no	yes	no
directory	no	no	yes	no
logging level	no	no	yes	no
TTL value	no	no	yes	no
display flag	no	no	yes	no
phase start	no	no	yes	no
last KSK rollover	no	no	yes	no
last ZSK rollover	no	no	yes	no

Table 9: **lsroll** Output Formats

OPTIONS

There are three types of options recognized by **lsroll**: record-selection options, attribute-selection options, and output-format options. Each type is described in the subsections below.

Record-selection Options

These options select the records that will be displayed by **lsroll**. By default, all records will be displayed; selecting one or the other of these options will restrict the records shown.

In order to simplify the **lsroll** code and keep it easily understandable, these options are mutually exclusive.

-roll

List all "roll" records in the *rollrec* file.

-skip

List all "skip" records in the *rollrec* file.

Attribute-selection Options

These options select the attributes of the records that will be displayed by **lsroll**.

-type

Include each *rollrec* record's type in the output. The type will be either "roll" or "skip".

-zone

The record's zonefile is included in the output. This field is part of the default output.

-keyrec

The record's *keyrec* file is included in the output. This field is part of the default output.

-kskphase

The record's KSK rollover phase are included in the output. If this option is given with the *-zskphase* option, then the output will follow the format described for the *-phases* option. This field is part of the default output.

-zskphase

The record's ZSK rollover phase are included in the output. If this option is given with the *-kskphase* option, then the output will follow the format described for the *-phases* option. This field is part of the default output.

-phases

The record's KSK and ZSK rollover phases are included in the output. The listing is given with the KSK phase first, followed by the ZSK phase.

Examples of output from this option are:

KSK Phase	**ZSK Phase**	**Output**
0	0	0/0
3	0	3/0
0	5	0/5

-admin

The record's administrator value is included in the output. If an administrator value is not included in a *rollrec*, then the value "(defadmin)" will be given.

-directory

The name of the directory that holds the zone's files is included in the output. If a zone directory is not included in a *rollrec*, then the value "(defdir)" will be given.

-loglevel

The **rollerd** logging level for this zone. This value may be given in the *rollrec* file in either the textual or numeric form. The textual form of the logging level will be displayed, not the numeric. If a logging level value is not included in a *rollrec*, then the value "(deflog)" will be given. If an undefined logging level value is included in a *rollrec*, then the value "(unknownlog)" will be given.

-ttl

The record's TTL value is included in the output.

-display

The record's display flag, used by **blinkenlights**, is included in the output.

-phstart

The record's rollover phase is included in the output. If no rollover has yet been performed for this zone, an empty date is given.

-lastksk

The record's last KSK rollover date is included in the output. If no KSK rollover has yet been performed for this zone, an empty date is given.

-lastzsk

The record's last ZSK rollover date is included in the output. If no ZSK rollover has yet been performed for this zone, an empty date is given.

Output-format Options

These options select the type of output that will be given by **lsroll**.

-count

Only a count of matching keyrecs in the *rollrec* file is given.

-headers

Display explanatory column headers.

-terse

Terse output is given. Only the record name and any other fields specifically selected are included in the output.

-long

Long output is given. All record fields are included.

-help

Display a usage message.

SEE ALSO

blinkenlights(8), rollchk(8), rollinit(8), rollerd(8)

Net::DNS::SEC::Tools::rollrec.pm(3)

Net::DNS::SEC::Tools::file-rollrec(5)

2.4.7 rollctl

NAME

rollctl - Send commands to the DNSSEC-Tools rollover daemon

SYNOPSIS

```
  rollctl [options]
```

DESCRIPTION

The **rollctl** command sends commands to the DNSSEC-Tools rollover daemon, **rollerd**. Multiple options may be specified on a single command line and they will be executed in *alphabetical* order. The exception to this ordering is that the *-shutdown* command will always be executed last.

In most cases, **rollerd** will send a response to **rollctl**. **rollctl** will print a success or failure message, as appropriate.

OPTIONS

The following options are handled by **rollctl**.

-display

Starts the rollover status GUI.

-dspub zone

Indicates that *zone*'s parent has published a new DS record for *zone*.

-dspuball

Indicates that DS records have been published for all zones in phase 6 of KSK rollover.

-halt

Cleanly halts **rollerd** execution.

-logfile logfile

Sets the **rollerd** log file to *logfile*. This must be a valid logging file, meaning that if *logfile* already exists, it must be a regular file. The only exceptions to this are if *logfile* is **/dev/stdout** or **/dev/tty**.

-loglevel loglevel

Sets the **rollerd** logging level to *loglevel*. This must be one of the valid logging levels defined in **rollmgr.pm(3)**.

-nodisplay

Stops the rollover status GUI.

-rollall

Initiates rollover for all the zones defined in the current *rollrec* file.

-rollrec rollrec_file

Sets the *rollrec* file to be processed by **rollerd** to *rollrec_file*.

-rollzone zone

Initiates rollover for the zone named by *zone*.

-runqueue

Wakes up **rollerd** and has it run its queue of *rollrec* entries.

-shutdown

Synonym for **-halt**.

-skipall

Stops rollover for all zones in the current *rollrec* file.

-skipzone zone

Stops rollover for the zone named by *zone*.

-sleeptime sleeptime

Sets **rollerd**'s sleep time to *sleeptime*. *sleeptime* must be an integer at least as large as the *$MIN_SLEEP* value in **rollerd**.

-status

Has **rollerd** write several of its operational parameters to its log file. The parameters are also reported to **rollctl**, which prints them to the screen.

-zonelog

Set the logging level for the specified zone. The new logging level is only for the current execution of **rollerd** and is not saved to the active *rollrec* file.

-zonestatus

Has **rollerd** write the status of zones in the current *rollrec* file to the **rollerd** log file. The status is also reported to **rollctl**, which prints it to the screen.

-quiet

Prevents output from being given. Both error and non-error output is stopped.

-help

Displays a usage message.

FUTURE

The following modifications may be made in the future:

command execution order

The commands will be executed in the order given on the command line rather than in alphabetical order.

SEE ALSO

Net::DNS::SEC::Tools::rollmgr.pm(3), Net::DNS::SEC::Tools::rollrec.pm(3)

rollerd(8)

2.4.8 rolllog

NAME

rolllog - DNSSEC-Tools utility to write messages to the DNSSEC rollover log file

SYNOPSIS

```
  rolllog -loglevel <level> <log_message>
```

DESCRIPTION

The **rolllog** program writes log messages to the DNSSEC rollover log file. **rolllog** does not actually write the messages itself; rather, it sends them to the **rollerd** rollover daemon to write the messages. **rollerd** keeps track of a logging level, and only messages of that level or higher are written to the log file.

OPTIONS

The following options are recognized:

-loglevel level

Logging level of this message. The valid levels are defined in **rollmgr.pm**(3). This option is required.

-help

Display a usage message.

-Version

Display a version message.

SEE ALSO

rollctl(8), rollerd(8)

Net::DNS::SEC::Tools::rollmgr.pm(3)

2.4.9 blinkenlights

NAME

blinkenlights - DNSSEC-Tools rollerd GUI

SYNOPSIS

```
  blinkenlights <rollrec-file>
```

DESCRIPTION

blinkenlights is a GUI tool for use with monitoring and controlling the DNSSEC-Tools **rollerd** program. It displays information on the current state of the zones **rollerd** is managing. The user may control some aspects of **rollerd**'s execution using **blinkenlights** menu commands.

blinkenlights creates a window in which to display information about each zone **rollerd** is managing. (These zones are those in **rollerd**'s current *rollrec* file.) As a zone's rollover status changes, **blinkenlights** will update its display for that zone. Skipped zones, zones listed in the *rollrec* file but which are not in rollover or normal operation, are displayed but have very little useful information to display.

The user may also select a set of zones to hide from the display. These zones, if in the rolling state, will continue to roll; however, their zone information will not be displayed. Display state for each zone will persist across **blinkenlights** executions.

Menu commands are available for controlling **rollerd**. The commands which operate on a single zone may be executed by keyboard shortcuts. The zone may be selected either by clicking in its "zone stripe" or by choosing from a dialog box. Display and execution options for **blinkenlights** are also available through menu commands. More information about the menu commands is available in the MENU COMMANDS section.

blinkenlights is only intended to be started by **rollerd**, not directly by a user. There are two ways to have **rollerd** start **blinkenlights**. First, **rollctl** may be given the *-display* option. Second, the *-display* option may be given on **rollerd**'s command line.

SCREEN LAYOUT

The **blinkenlights** window is laid out as a series of "stripes". The top stripe contains status information about **rollerd**, the second stripe contains column headers, and the bulk of the window consists of zone stripes. The list below provides more detail on the contents of each stripe.

See the WINDOW COLORS section for a discussion of the colors used for the zone stripes.

- **rollerd** information stripe

  The information stripe contains four pieces of information: **rollerd**'s current *rollrec* file, the count of rolling zones, the count of skipped zones, and the amount of time

rollerd waits between processing its queue. Coincidentally, that last datum is also the amount of time between **blinkenlights** screen updates.

- column headers stripe

  This stripe contains the column headers for the columns of each zone stripe.

- zone stripes

  Each zone managed by **rollerd** (i.e., every zone in the current *rollrec* file) will have a zone stripe which describes that zone's current state. The stripe is divided into four sections: the zone name, the current rollover state, and the zone's DNSSEC keys.

  The zone name section just contains the name of the zone.

  The rollover state section contains the rollover phase number, a text explanation of the phase, and the amount of time remaining in that rollover phase. The phase explanation is "normal operation" when the zone isn't currently in rollover.

  The DNSSEC key section contains two subsections, one for the zone's ZSK keys and another for the zone's KSK keys. Each subsection contains the names of the signing sets active for the zone. The ZSK subsection lists the Current, Published, and New ZSK keys; the KSK subsection lists the Current and Published.

  See the WINDOW COLORS section for a discussion of the colors used for the zone stripes.

WINDOW COLORS

The default **blinkenlights** configuration uses window coloring to provide visual cues and to aid in easily distinguishing zone information. The default window coloring behavior gives each zone stripe has its own color and the rollover state section of each zone stripe is shaded to show the zone's phase. Window coloring can be turned off (and on) with configuration options and menu commands.

Color Usage

The two window coloring behaviors are discussed more fully below:

- zone stripe colors

  Each rolling zone's stripe is given one of three colors: blue, red, or green. The color is assigned on a top-down basis and the colors wrap if there are more than three zones. So, the first zone is always blue, the second zone red, the third zone green, the fourth zone blue, etc.

  The colors do not stay with a particular zone. If a rolling zone becomes a skipped zone, the zone stripes will be reassigned new colors to account for that skipped zone.

  Skipped zones are not colored with these three colors. Stripes for skipped zones are colored either grey or a color set in the configuration file. If you choose to use a non-standard color for skipped zones your should ensure that it is **not** one of the colors used for rolling zones' stripes. Modifying the *skipcolor* configuration field allows the skipped-zone color to be changing.

The *colors* configuration field can be used to turn on or off the use of colors for zone stripes. If stripe coloring is turned off, then every stripe will be displayed using the *skipcolor* color.

- rollover-state shading

  The only portion of a zone stripe that changes color is the status column; the color of the rest of the zone stripe stays constant. Before a zone enters rollover, the status column is the same color as the rest of the stripe. When the zone enters rollover, the status column's color is changed to a very light shade of the stripe's normal color. As the rollover phases progress towards rollover completion, the status column's shade darkens. Once rollover completes, the status column returns again to the same shade as the rest of that stripe.

  The *shading* configuration field can be used to turn on or off the use of shading in the rollover-state column. If shading is turned off, then the zone stripe will be a solid color.

  See the CONFIGURATION FILE section for information on setting the configuration fields.

Colors Used

The color names are taken from the X11 **rgb.txt** file (X11 1.1.3 - XFree86 4.4.0 for MacOS X.) If these aren't available in your **rgb.txt** file, similar names should be selected. The actual red/green/blue values used are given below to assist in finding suitable replacements. These values were taken from the **rgb.txt** file.

Blue Shades:

Color Name	**Red Value**	**Green Value**	**Blue Value**
blue	0	0	255
lightblue2	178	223	238
darkslategray1	151	255	255
skyblue1	135	206	255
steelblue1	99	184	255
turquoise1	0	245	255
cornflower blue	100	149	237
dodger blue	30	144	255

Table 10: Blue Shades

Red Shades:

Color Name	**Red Value**	**Green Value**	**Blue Value**
red	255	0	0
pink	255	192	203
lightsalmon1	255	160	122
tomato	255	99	71
indianred	205	92	92
violetred1	255	62	150
orangered1	255	69	0
firebrick1	255	48	48

Table 11: Red Shades

Green Shades:

Color Name	**Red Value**	**Green Value**	**Blue Value**
green	0	255	0
darkseagreen1	193	255	193
darkolivegreen1	202	255	112
lightgreen	144	238	144
seagreen1	84	255	159
spring green	0	255	127
greenyellow	173	255	47
lawngreen	124	252	0

Table 12: Green Shades

MENU COMMANDS

A number of menu commands are available to control the behavior of **blinkenlights** and to send commands to **rollerd**. These commands are discusses in this section.

File Menu

The commands in this menu are basic GUI commands.

- Quit

  blinkenlights will stop execution.

Options Menu

The commands in this menu control the appearance and behavior of **blinkenlights**.

- Row Colors (toggle)

  This menu item is a toggle to turn on or off the coloring of zone stripes. If row coloring is turned off, zone stripes will all be the same color. If row coloring is turned on, zone stripes will be displayed in varying colors. See the WINDOW COLORS section for a discussion of row coloring.

- Status Column Shading (toggle)

  This menu item is a toggle to turn on or off the shading of the zone status column. If shading is turned off, the zone stripes will present a solid, unchanging band of color for each zone. If shading is turned on, the color of the zone status column will change according to the zone's rollover state.

- Skipped Zones Display (toggle)

  This menu item is a toggle to turn on or off the display of skipped zones. If display is turned off, zone stripes for skipped zones will not be displayed. If display is turned on, zone stripes for all zones will be displayed.

- Modification Commands (toggle)

  In some situations, it may be desirable to turn off **blinkenlights**' ability to send commands to **rollerd**. This menu item is a toggle to turn on or off this ability. If the commands are turned off, then the "Zone Control" menu and keyboard shortcuts are disabled. If the commands are turned on, then the "Zone Control" menu and keyboard shortcuts are enabled.

- Font Size

  This menu item allows selection of font size of text displayed in the main window.

  Normally, changing the font size causes the window to grow and shrink as required. However, on Mac OS X there seems to be a problem when the size selected increases the window size to be greater than will fit on the screen. If the font size is subsequently reduced, the window size does not shrink in response.

General Control Menu

The commands in this menu are GUI interfaces for the **rollctl** commands related to *general* zone management.

- Roll Selected Zone

  The selected zone will be moved to the rollover state. This only has an effect on skipped zones. A zone may be selected by clicking on its zone stripe. If this command is selected without a zone having been selected, a dialog box is displayed from which a currently skipped zone may be chosen.

- Roll All Zones

  All zones will be moved to the rollover state. This has no effect on currently rolling zones.

- Run the Queue

  rollerd is awoken and runs through its queue of zones. The operation required for each zone is then performed.

- Skip Selected Zone

  The selected zone will be moved to the skipped state. This only has an effect on rolling zones. A zone may be selected by clicking on its zone stripe. If this command is selected without a zone having been selected, a dialog box is displayed from which a currently rolling zone may be chosen.

- Skip All Zones

  All zones will be moved to the skipped state. This has no effect on currently skipped zones.

- Halt Rollerd

  rollerd's execution is halted. As a result, **blinkenlights**' execution will also be halted.

KSK Control Menu

The commands in this menu are GUI interfaces for the **rollctl** commands related to KSK-specific zone management.

- DS Published Selected Zone

  This command is used to indicate that the selected zone's parent has published a new DS record for the zone. It moves the zone from phase 6 to phase 7 of KSK rollover.

- DS Published All Zones

  This command is used to indicate that all the zones in KSK rollover phase 6 have new DS records published by their parents. It moves all these zones from phase 6 to phase 7 of KSK rollover.

Zone Display Menu

The commands in this menu are GUI interfaces parts of the zone display. There are commands for displaying and hiding both zone stripes and key columns. The commands allow all, some, or none of the zone stripes and key columns to be displayed. Undisplayed rolling zones will continue to roll, but they will do so without the **blinkenlights** window indicating this.

- Zone Selection

  A dialog box is created that holds a list of the zones currently managed by **rollerd**. The user may select which zones should be displayed by clicking on the zone's checkbox. Zones with a selected checkbox will be displayed; zones without a selected checkbox will not be displayed.

- Display All Zones

  All zones will be displayed in the **blinkenlights** window.

- Hide All Zones

  No zones will be displayed in the **blinkenlights** window.

- KSK Sets (toggle)

  This menu item is a toggle to turn on or off the display of KSK signing set names. If display is turned off, the columns holding the KSK signing set names and labels will be removed from the display and the display window will shrink. If display is turned on, the columns holding the KSK signing set names and labels will be restored to the display and the display window will be expanded.

  When displayed, KSK signing sets will always be the right-most columns.

- ZSK Sets (toggle)

  This menu item is a toggle to turn on or off the display of ZSK signing set names. If display is turned off, the columns holding the ZSK signing set names and labels will be removed from the display and the display window will shrink. If display is turned on, the columns holding the ZSK signing set names and labels will be restored to the display and the display window will be expanded.

  When displayed, ZSK signing sets will always be immediately to the right of the zone status column.

- Hide All Keysets

  Turns off display of the KSK and ZSK signing set names.

- Show All Keysets

  Turns on display of the KSK and ZSK signing set names.

Help Menu

The commands in this menu provide assistance to the user.

- Help

  Display a window containing help information.

CONFIGURATION FILE

Several aspects of **blinkenlights**' behavior may be controlled from configuration files. Configuration value may be specified in the DNSSEC-Tools configuration file or in a more specific **rc.blinkenlights**. The system-wide **blinkenlights** configuration file is in the DNSSEC-Tools configuration directory and is named **blinkenlights.conf**. Multiple **rc.blinkenlights** files may exist on a system, but only the one in the directory in which **blinkenlights** is executed is used.

The following are the available configuration values:

The **rc.blinkenlights** file is **only** searched for in the directory in which **blinkenlights** is executed. The potential problems inherent in this may cause these **blinkenlights**-specific configuration files to be removed in the future.

This file is in the "field value" format, where *field* specifies the output aspect and *value* defines the value for that field. The following are the recognized fields:

Configuration Value	**Meaning**
colors	Turn on/off use of colors on zone stripes
fontsize	The size of the font in the output window
modify	Turn on/off execution of rollerd modification commands
shading	Turn on/off shading of the status columns
showskip	Turn on/off display of skipped zones
skipcolor	The background color used for skipped zones

Table 13: **blinkenlights** Configuration File Entries

Empty lines and comments are ignored. Comment lines are lines that start with an octothorpe ('#').

Spaces are not allowed in the configuration values.

Choose your skipcolors carefully. The only foreground color used is black, so your background colors must work well with black.

REQUIREMENTS

blinkenlights is implemented in Perl/Tk, so both Perl and Perl/Tk must be installed on your system.

WARNINGS

blinkenlights has several potential problems that must be taken into account.

development environment

blinkenlights was developed and tested on a single-user system running X11. While it works fine in this environment, it has not been run on a system with many users or in a situation where the system console hasn't been in use by the **blinkenlights** user.

long-term performance issues

In early tests, the longer **blinkenlights** runs, the slower the updates become. This is *probably* a result of the Tk implementation or the way Tk interfaces with X11. This is pure supposition, though.

This performance impact is affected by a number of things, such as the number of zones managed by **rollerd** and the length of **rollerd**'s sleep interval. Large numbers of zones or very short sleep intervals will increase the possibility of **blinkenlights**' performance degrading.

This appears to have been resolved by periodically performing a complete rebuild of the screen. **blinkenlights** keeps track of the number of screen updates it makes and rebuilds the screen when this count exceeds a threshold. The threshold is built into **blinkenlights** and stored in the *$PAINTMAX* variable. This threshold may be adjusted if there are too many screen rebuilds or if **blinkenlights**' performance slows too much. Raising the number will reduce the screen rebuilds; lowering the number will (may) increase performance.

SEE ALSO

rollctl(8), rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::timetrans(3)

Net::DNS::SEC::Tools::keyrec(5), Net::DNS::SEC::Tools::rollrec(5),

3 DNSSEC Library Routines

Several libraries have been developed to provide DNSSEC-validated resolution, translation, and querying services. These DNSSEC-Tools libraries are:

libsres - Secure Resolver Library

libsres provides the resolver component of the "validating resolver". It is capable of recursively obtaining answers for an application (validator) from a DNSSEC-aware name server. Resolver policy will eventually be used to control the flags (CD, RD etc) that are sent in the query to the name servers, as well as other parameters, such as the name server to which the query is to be sent.

This library provides very basic functionality for name resolution. The data structures and interfaces exported to applications are still in a state of flux and are expected to change. Many corner cases are still not supported.

libval - DNSSEC validation library

libval provides DNSSEC resource-record validation functionality. It relies on the resolver component to fetch answers from a DNSSEC-aware name server.

As of now there is no functionality to traverse the chain-of-trust while performing record validation and also no support for validation policies. As such, the interfaces defined herein are in a state of flux and are expected to change.

val_getaddrinfo()

Library routines to retrieve DNSSEC-validated network address and service translations.

val_gethostbyname()

Library routines to perform DNSSEC validation of DNS queries.

val_query()

Library routines to do DNSSEC-validated resolution of DNS queries.

This section contains man pages describing these libraries.

3.0.10 libsres

NAME

query_send(), *response_rcv()*, *get()* - send queries and receive responses from a DNS name server

clone_ns(), *clone_ns_list()*, *free_name_server()*, *free_name_servers()* - manage name server lists

print_response() - display answers returned from the name server

SYNOPSIS

```
    #include <resolver.h>

    int query_send(const char          *name,
                   const u_int16_t     type,
                   const u_int16_t     class,
                   struct name_server  *nslist,
                   int                 edns0_size,
                   int                 *trans_id);

    int response_recv(int                 *trans_id,
                      struct name_server  **respondent,
                      u_int8_t            **response,
                      u_int32_t           *response_length);

    int get(const char          *name_n,
            const u_int16_t     type_h,
            const u_int16_t     class_h,
            struct name_server  *nslist,
            int                 edns0_size,
            struct name_server  **respondent,
            u_int8_t            **response,
            u_int32_t           *response_length);

    int clone_ns(struct name_server **cloned_ns,
                 struct name_server *ns);

    int clone_ns_list(struct name_server **ns_list,
                      struct name_server *orig_ns_list);

    void free_name_server(struct name_server **ns);

    void free_name_servers(struct name_server **ns);

    void print_response(u_int8_t *response, int response_length);
```

DESCRIPTION

The *query_send()* function sends a query to the name servers specified in *nslist*. The query is comprised of the <*name, class, type*> tuple and *trans_id* provides a handle to this transaction within the *libsres* library. The buffer size advertised in the EDNS0 option can be set using the *ends0_size* argument.

The *response_recv()* function returns the answers, if available, from the name server that responds for the query identified by *trans_id*. The response is available in *response* and the responding name server is returned in *respondent*. The length of the response in bytes is returned in *response_length*.

The *get()* function provides a wrapper around the *query_send()* and *response_recv()* functions. After sending a request, it blocks until a response is received from some name server or until the request times out. The *libsres* library does not automatically follow referrals; responses containing referrals are treated as valid responses.

The memory pointed to by **respondent* is internally allocated by the *libsres* library and must be freed by the invoker using *free_name_server()*. An entire list of name servers can be freed using *free_name_servers()*. A copy of the name server can be created using *clone_ns()* and a copy of a name server list can be made using *clone_ns_list()*.

print_response() provides a convenient way to display answers returned in *response* by the name server.

The *name_server* structure is defined in **resolver.h** as follows:

```
    struct name_server
    {
        u_int8_t ns_name_n[NS_MAXCDNAME];
        void *ns_tsig;
        u_int32_t ns_security_options;
        u_int32_t ns_status;
        u_long  ns_options;
        int ns_retry;
        int ns_retrans;
        struct name_server *ns_next;
        int ns_number_of_addresses;
        struct sockaddr_storage **ns_address;
    }
;
```

ns_name_n

The name of the zone for which this name server is authoritative.

ns_tsig

The *tsig* key that should be used to protect messages sent to this name server. This field is currently unused and must be set to NULL.

ns_security_options

The security options for the zone. This field is currently unused and must be set to **ZONE_USE_NOTHING**.

ns_status

The status of the zone. This field indicates how the zone information was obtained. The invoker must set this value to **SR_ZI_STATUS_UNSET**. Zone information obtained through referrals have a value of **SR_ZI_STATUS_LEARNED** for this field.

ns_options

Specifies additional resolver flags. Currently defined flags are **RES_RECURSE**, which sets the "Recursion Desired" flag; **RES_USE_DNSSEC**, which sets the "DNSSEC OK" bit in the EDNS0 header; and **RES_DEBUG**, which enables debugging.

ns_retry

Specifies the maximum number of attempts that must be made to obtain a name from an unresponsive name server before giving up.

ns_retrans

Specifies the retransmission interval in seconds for queries sent to unresponsive name servers.

ns_next

The address of the next name server in the list.

ns_number_of_addresses

The number of elements in the array *ns_addresses*. This field is currently unused.

ns_addresses

The IP address of the name server.

OTHER SYMBOLS EXPORTED

The *libsres* library also exports the following BIND functions, documentation for which can be found in the BIND sources and documentation manuals:

```
res_nametoclass
res_nametotype
ns_name_ntop
ns_name_pton
ns_name_unpack
ns_parse_ttl
p_class
p_section
p_type
```

The *p_type()* function exported from *libsres* has been augmented such that it recognizes the various DNSSEC type codes such DNSKEY, RRSIG, NSEC, NSEC3 and DLV.

RETURN VALUES

SR_UNSET

No error.

SR_CALL_ERROR

An invalid parameter was passed to *get()*, *query_send()*, or *response_recv()*.

SR_INTERNAL_ERROR

The resolver encountered some internal error.

SR_TSIG_ERROR

The resolver encountered some TSIG-related error. This is currently not implemented.

SR_NO_ANSWER

No answers were received from any name server.

SR_NO_ANSWER_YET

No answer currently available; the query is still active.

SR_HEADER_ERROR

The length and count of records in the header were incorrect.

SR_NXDOMAIN

The queried name did not exist.

SR_FORMERR

The name server was not able to parse the query message.

SR_SERVFAIL

The name server was not reachable.

SR_NOTIMPL

A particular functionality is not yet implemented.

SR_REFUSED

The name server refused to answer this query.

SR_DNS_GENERIC_FAILURE

Other failure returned by the name server and reflected in the returned message **RCODE**.

SR_EDNS_VERSION_ERROR

The EDNS version was not recognized.

SR_NAME_EXPANSION_FAILURE

A failure was encountered while trying to expand a compressed domain name.

CURRENT STATUS

There is currently no support for IPv6.

There is limited support for specifying resolver policy; members of the *name_server* structure are still subject to change.

SEE ALSO

libval(3)

3.0.11 libval

NAME

val_create_context(), *val_create_context_with_conf()*, *val_free_context()* - manage validator contex

val_resolve_and_check(), *val_free_result_chain()* - query and validate answers from a DNS name server

val_istrusted() - check if status value corresponds to that of a trustworthy answer

val_isvalidated() - check if status value represents an answer tha cryptographically chains down from a configured trust anchor

val_does_not_exist() - check if status value represents one of the non-existence types

p_val_status(), *p_ac_status()*, *p_val_error()* - display validation status, authentication chain status and error information

dnsval_conf_get(), *resolv_conf_get()*, *root_hints_get()* - get the default location for the validator configuration files

dnsval_conf_set(), *resolv_conf_set()*, *root_hints_set()* - set the default location for the validator configuration files

SYNOPSIS

```
    #include <validator.h>

    int val_create_context(const char *label, val_context_t **newcontext);

    int val_create_context_with_conf(char *label,
                                     char *dnsval_conf,
                                     char *resolv_conf,
                                     char *root_conf,
                                     val_context_t ** newcontext);

    int val_resolve_and_check(val_context_t *context,
                              u_char *domain_name_n,
                              const u_int16_t class,
                              const u_int16_t type,
                              const u_int8_t  flags,
                              struct val_result_chain  **results);

    char *p_val_status(val_status_t valerrno);

    char *p_ac_status(val_astatus_t auth_chain_status);

    char *p_val_error(int err);
```

```
int val_istrusted(val_status_t val_status);

int val_isvalidated(val_status_t val_status);

int val_does_not_exist(val_status_t status);

void val_free_result_chain(struct val_result_chain *results);

void val_free_context(val_context_t *context);

char *resolv_conf_get(void);

int resolv_conf_set(const char *name);

char *root_hints_get(void);

int root_hints_set(const char *name);

char *dnsval_conf_get(void);

int dnsval_conf_set(const char *name);
```

DESCRIPTION

The *val_resolve_and_check()* function queries a set of name servers for the <*domain_name_n, type, class*> tuple and to verifies and validates the response. Verification involves checking the RRSIGs, and validation is verification of an authentication chain from a configured trus anchor. The *domain_name_n* parameter is the queried name in DNS wire format. The conversion from host format to DNS wire format can be done using the *ns_name_pton()* function exported by the *libsres(3)* library.

The *flags* parameter can be used to control the results of validation. Two values, which may be ORed together, are currently defined for this field. **VAL_QUERY_DONT_VALIDATE** causes the validator to disable validation for this query. **VAL_QUERY_NO_DLV** causes the validator to disable DLV processing for this query. The second flag is only available if the *libval(3)* library has been compiled with DLV support.

The first parameter to *val_resolve_and_check()* is the validator context. Applications can create a new validator context using the *val_create_context()* function. This function parses the resolver and validator configuration files and creates the handle *newcontext* to this parsed information. Information stored as part of validator context includes the validation policy and resolver policy.

Validator and resolver policies are read from the **/etc/dnsval.conf** and **/etc/resolv.conf** files by default. **/etc/root.hints** provides bootstrapping information for the validator when it functions as a full resolver (see **dnsval.conf(3)**). The default locations of each of these files may be changed using the interfaces *dnsval_conf_set()*, *resolv_conf_set()*, and *root_hints_set()*, respectively. The corresponding "get" interfaces (namely *dnsval_conf_get()*, *resolv_conf_get()*, and *root_hints_get()*) can be used to return the current location from where these configuration files are read. The configuration file locations may also be specified on a per-context basis using the *val_create_context_with_conf()* function.

Answers returned by *val_resolve_and_check()* are made available in the **results* linked list. Each answer corresponds to a distinct RRset; multiple RRs within the RRset are part of the same answer. Multiple answers are possible when *type* is *ns_t_any* or *ns_t_rrsig*.

Individual elements in **results* point to *val_authentication_chain* linked lists. The authentication chain elements in *val_authentication_chain* contain the actual RRsets returned by the name server in response to the query.

Validation result values returned in *val_result_chain* and authentication chain status values returned in each element of the *val_authentication_chain* linked list can be can be converted into ASCII format using the *p_val_status()* and *p_ac_status()* functions respectively.

While some applications such as DNSSEC troubleshooting utilities and packet inspection tools may look at individual authentication chain elements to identify the actual reasons for validation error, most applications will only be interested in a single error code for determining the authenticity of data.

val_isvalidated() identifies if a given validation result status value corresponds to an answer that was cryptographically verified and validated using a locally configured trust anchor.

val_istrusted() identifies if a given validator status value is trusted. An answer may be locally trusted without being validated.

val_does_not_exist() identifies if a given validator status value corresponds to one of the non-existence types.

The *libval* library internally allocates memory for **results* and this must be freed by the invoking application using the *free_result_chain()* interface.

DATA STRUCTURES

struct val_result_chain

```
struct val_result_chain
{
    val_status_t                    val_rc_status;
    struct val_authentication_chain *val_rc_answer;
    int                             val_rc_proof_count;
    struct val_authentication_chain *val_rc_proofs[MAX_PROOFS];
    struct val_result_chain         *val_rc_next;
};
```

val_rc_answer

Authentication chain for a given RRset.

val_rc_next

Pointer to the next RRset in the set of answers returned for a query.

val_rc_proofs

Pointer to authentication chains for any proof of non-existence that were returned for the query.

val_rc_proof_count

Number of proof elements stored in *val_rc_proofs*. The number canno exceed **MAX_PROOFS**.

val_rc_status

Validation status for a given RRset. This can be one of the following:

VAL_SUCCESS

Answer received and validated successfully.

VAL_LOCAL_ANSWER

Answer was available from a local file.

VAL_BARE_RRSIG

No DNSSEC validation possible, query was for an RRSIG.

VAL_NONEXISTENT_NAME

No name was present and a valid proof of non- existence confirming the missing name (NSEC or NSEC3 span) was returned. The components of the proof were also individually validated.

VAL_NONEXISTENT_TYPE

No type exists for the name and a valid proof of non-existence confirming the missing name was returned. The components of the proof were also individually validated.

VAL_NONEXISTENT_NAME_NOCHAIN

No name was present and a valid proof of non- existence confirming the missing name was returned. The components of the proof were also identified to be trustworthy, but they were not individually validated.

VAL_NONEXISTENT_TYPE_NOCHAIN

No type exists for the name and a valid proof of non-existence confirming the missing name (NSEC or NSEC3 span) was returned. The components of the proof were also identified to be trustworthy, but they were not individually validated.

VAL_DNS_CONFLICTING_ANSWERS

Multiple conflicting answers received for a query.

VAL_DNS_QUERY_ERROR

Some error was encountered while sending the query.

VAL_DNS_RESPONSE_ERROR

No response returned or response with an error rcode value returned.

VAL_DNS_WRONG_ANSWER

Wrong answer received for a query.

VAL_DNS_REFERRAL_ERROR

Some error encountered while following referrals.

VAL_DNS_MISSING_GLUE

Glue was missing.

VAL_BOGUS_PROVABLE
Response could not be validated due to signature verification failures or the inability to verify proofs for exactly one component in the authentication chain below the trust anchor.

VAL_BOGUS
Response could not be validated due to signature verification failures or the inability to verify proofs for an indeterminate number of components in the authentication chain.

VAL_NOTRUST
All available components in the authentication chain verified properly, but there was no trust anchor available.

VAL_IGNORE_VALIDATION
Local policy was configured to ignore validation for the zone from which this data was received.

VAL_TRUSTED_ZONE
Local policy was configured to trust any data received from the given zone.

VAL_UNTRUSTED_ZONE
Local policy was configured to reject any data received from the given zone.

VAL_PROVABLY_UNSECURE
The record or some ancestor of the record in the authentication chain towards the trust anchor was known to be provably unsecure.

VAL_BAD_PROVABLY_UNSECURE
The record or some ancestor of the record in the authentication chain towards the trust anchor was known to be provably unsecure. But the provably unsecure condition was configured as untrustworthy.

Status values in *val_status_t* returned by the validator can be displayed in ASCII format using *p_val_status()*.

struct val_authentication_chain

```
struct val_authentication_chain
{
    val_astatus_t                    val_ac_status;
    struct val_rrset                *val_ac_rrset;
    struct val_authentication_chain *val_ac_trust;
};
```

val_ac_status

Validation state of the authentication chain element. This field will contain the status code for the given component in the authentication chain. This field may contain one of the following values:

VAL_AC_UNSET
The status was not set.

VAL_AC_DATA_MISSING
No data were returned for a query and the DNS did not indicate an error.

VAL_AC_RRSIG_MISSING
RRSIG data could not be retrieved for a resource record.

VAL_AC_DNSKEY_MISSING
The DNSKEY for an RRSIG covering a resource record could not be retrieved.

VAL_AC_DS_MISSING
The DS record covering a DNSKEY record was not available.

VAL_AC_NOT_VERIFIED
All RRSIGs covering the RRset could not be verified.

VAL_AC_VERIFIED
At least one RRSIG covering a resource record had a status of **VAL_AC_RRSIG_VERIFIED**.

VAL_AC_TRUST_KEY
A given DNSKEY or a DS record was locally defined to be a trust anchor.

VAL_AC_IGNORE_VALIDATION
Validation for the given resource record was ignored, either because of some local policy directive or because of some protocol-specific behavior.

VAL_AC_TRUSTED_ZONE
Local policy defined a given zone as trusted, with no further validation being deemed necessary.

VAL_AC_UNTRUSTED_ZONE
Local policy defined a given zone as untrusted, with no further validation being deemed necessary.

VAL_AC_PROVABLY_UNSECURE
The authentication chain from a trust anchor to a given zone could no be constructed due to the provable absence of a DS record for this zone in the parent.

VAL_AC_BARE_RRSIG
The response was for a query of type RRSIG. RRSIGs contain the cryptographic signatures for other DNS data and cannot themselves be validated.

VAL_AC_NO_TRUST_ANCHOR
There was no trust anchor configured for a given authentication chain.

VAL_AC_DNS_CONFLICTING_ANSWERS
Multiple conflicting answers received for a query.

VAL_AC_DNS_QUERY_ERROR
Some error was encountered while sending the query.

VAL_AC_DNS_RESPONSE_ERROR
No response returned or response with an error rcode value returned.

VAL_AC_DNS_WRONG_ANSWER
Wrong answer received for a query.

VAL_AC_DNS_REFERRAL_ERROR
Some error encountered while following referrals.

VAL_AC_DNS_MISSING_GLUE
Glue was missing.

val_ac_rrset

Pointer to an RRset of type *val_rrset* obtained from the DNS response.

val_ac_trust

Pointer to an authentication chain element that either contains a DNSKEY RRset that can be used to verify RRSIGs over the current record, or contains a DS RRset that can be used to build the chain-of-trust towards a trust anchor.

struct val_rrset

```
struct val_rrset
{
    u_int8_t      *val_msg_header;
    u_int16_t      val_msg_headerlen;
    u_int8_t      *val_rrset_name_n;
    u_int16_t      val_rrset_class_h;
    u_int16_t      val_rrset_type_h;
    u_int32_t      val_rrset_ttl_h;
    u_int32_t      val_rrset_ttl_x;
    u_int8_t       val_rrset_section;
    struct rr_rec *val_rrset_data;
    struct rr_rec *val_rrset_sig;
};
```

val_msg_header

Header of the DNS response in which the RRset was received.

val_msg_headerlen

Length of the header information in *val_msg_header*.

val_rrset_name_n

Owner name of the RRset represented in on-the-wire format.

val_rrset_class_h

Class of the RRset.

val_val_rrset_type_h

Type of the RRset.

val_rrset_ttl_h

TTL of the RRset.

val_rrset_ttl_x

The time when the TTL for this RRset is set to expire.

val_rrset_section

Section in which the RRset was received. This value may be **VAL_FROM_ANSWER**, **VAL_FROM_AUTHORITY**, or **VAL_FROM_ADDITIONAL**.

val_rrset_data

Response RDATA.

val_rrset_sig

Any associated RRSIGs for the RDATA returned in *val_rrset_data*.

struct rr_rec

```
struct rr_rec
{
     u_int16_t        rr_rdata_length_h;
     u_int8_t        *rr_rdata;
     val_astatus_t    rr_status;
     struct rr_rec   *rr_next;
};
```

rr_rdata_length_h

Length of data stored in *rr_rdata*.

rr_rdata

RDATA bytes.

rr_status

For each signature *rr_rec* member within the authentication chain *val_ac_rrset*, the validation status stored in the variable *rr_status* can return one of the following values:

VAL_AC_RRSIG_VERIFIED
The RRSIG verified successfully.

VAL_AC_WCARD_VERIFIED
A given RRSIG covering a resource record shows that the record was wildcard expanded.

VAL_AC_RRSIG_VERIFIED_SKEW
The RRSIG verified successfully after clock skew was taken into account.

VAL_AC_WCARD_VERIFIED_SKEW
A given RRSIG covering a resource record shows that the record was wildcard expanded, but it was verified only after clock skew was taken into account.

VAL_AC_RRSIG_VERIFY_FAILED
A given RRSIG covering an RRset was bogus.

VAL_AC_DNSKEY_NOMATCH
An RRSIG was created by a DNSKEY that did not exist in the apex keyset.

VAL_AC_RRSIG_ALGORITHM_MISMATCH
The keytag referenced in the RRSIG matched a DNSKEY but the algorithms were different.

VAL_AC_WRONG_LABEL_COUNT
The number of labels on the signature was greater than the count given in the RRSIG RDATA.

VAL_AC_BAD_DELEGATION
An RRSIG was created with a key that did not exist in the parent DS record set.

VAL_AC_RRSIG_NOTYETACTIVE
The RRSIG's inception time is in the future.

VAL_AC_RRSIG_EXPIRED
The RRSIG had expired.

VAL_AC_INVALID_RRSIG
The RRSIG could not be parsed.

VAL_AC_ALGORITHM_NOT_SUPPORTED
The RRSIG algorithm was not supported.

For each *rr_rec* member of type DNSKEY (or DS, where relevant) within the authentication chain *val_ac_rrset*, the validation status is stored in the variable *rr_status* and can return one of the following values:

VAL_AC_SIGNING_KEY
This DNSKEY was used to create an RRSIG for the resource record set.

VAL_AC_VERIFIED_LINK
This DNSKEY provided the link in the authentication chain from the trust anchor to the signed record.

VAL_AC_UNKNOWN_DNSKEY_PROTOCOL
The DNSKEY protocol number was unrecognized.

VAL_AC_UNKNOWN_ALGORITHM_LINK
This DNSKEY provided the link in the authentication chain from the trust anchor to the signed record, but the DNSKEY algorithm was unknown.

VAL_AC_INVALID_KEY
The key used to verify the RRSIG was not a valid DNSKEY.

VAL_AC_ALGORITHM_NOT_SUPPORTED
The DNSKEY or DS algorithm was not supported.

rr_next

Points to the next resource record in the RRset.

RETURN VALUES

Return values for various functions are given below. These values can be displayed in ASCII format using the *p_val_error()* function.

val_resolve_and_check()

VAL_NO_ERROR
No error was encountered.

VAL_GENERIC_ERROR
Generic error encountered.

VAL_NOT_IMPLEMENTED
Functionality not yet implemented.

VAL_BAD_ARGUMENT
Bad arguments passed as parameters.

VAL_INTERNAL_ERROR
Encountered some internal error.

VAL_NO_PERMISSION
No permission to perform operation. Currently not implemented.

VAL_RESOURCE_UNAVAILABLE
Some resource (crypto possibly) was unavailable. Currently not implemented.

val_create_context() and *val_create_context_with_conf()*

VAL_NO_ERROR
No error was encountered.

VAL_RESOURCE_UNAVAILABLE
Could not allocate memory.

VAL_CONF_PARSE_ERROR
Error in parsing some configuration file.

VAL_CONF_NOT_FOUND
A configuration file was not available.

VAL_NO_POLICY
The policy identifier being referenced was invalid.

FILES

The validator library reads configuration information from two files, **/etc/resolv.conf** and **/etc/dnsval.conf**.

See **dnsval.conf(5)** for a description of syntax for these files.

SEE ALSO

libsres(3)

dnsval.conf(5)

3.0.12 val_getaddrinfo()

NAME

val_getaddrinfo(), *val_freeaddrinfo()* - get DNSSEC-validated network address and service translation

SYNOPSIS

```
    #include <validator.h>

    int val_getaddrinfo(const struct val_context *ctx,
                        const char *nodename,
                        const char *servname,
                        const struct addrinfo *hints,
                        struct val_addrinfo **res,
                        val_status_t * val_status);

    void val_freeaddrinfo(struct val_addrinfo *ainfo);

    int val_getnameinfo(val_context_t * ctx,
                        const struct sockaddr *sa,
                        socklen_t salen,
                        char *host,
                        size_t hostlen,
                        char *serv,
                        size_t servlen,
                        int flags,
                        val_status_t * val_status);
```

DESCRIPTION

val_getaddrinfo() and *val_getnameinfo* perform DNSSEC validation of DNS queries. They are intended to be DNSSEC-aware replacements for *getaddrinfo(3)* and *getnameinfo(3)*.

val_getaddrinfo() returns a network address value of type *val_addrinfo* structure in the *res* parameter. *val_getnameinfo* is used to convert a *sockaddr* structure to a pair of host name and service strings.

```
    struct val_addrinfo
    {
            int ai_flags;
            int ai_family;
            int ai_socktype;
            int ai_protocol;
            size_t ai_addrlen;
            struct sockaddr *ai_addr;
            char * ai_canonname;
            struct val_addrinfo *ai_next;
            val_status_t  ai_val_status;
    };
```

The *val_addrinfo* structure is similar to the *addrinfo* structure (described in *getaddrinfo(3)*). *val_addrinfo* has an additional field *ai_val_status* that represents the validation status for that particular record. *val_status* gives the combined validation status value for all answers returned by the each of the functions. *val_istrusted()* and *val_isvalidated()* can be used to determine the trustworthiness of data and *p_val_status()* can be used to display the status value to the user in ASCII format (See *libval(3)* more for information).

The *ctx* parameter specifies the validation context, which can be set to NULL for default values (see *libval(3)* and **dnsval.conf** for more details on validation contexts and validation policy).

The *nodename*, *servname*, and *hints* parameters have similar syntax and semantics as the corresponding parameters for the original *getaddrinfo()* function. The *res* parameter is similar to the *res* parameter for *getaddrinfo()*, except that it is of type struct *val_addrinfo* instead of struct *addrinfo*. Please see the manual page for *getaddrinfo(3)* for more details about these parameters.

val_freeaddrinfo() frees the memory allocated for a *val_addrinfo* linked list.

RETURN VALUES

The *val_getaddrinfo()* function returns 0 on success and a non-zero error code on failure. **res* will point to a dynamically allocated linked list of *val_addrinfo* structures on success and will be NULL if no answer was available.

The *val_status* parameter gives an indication for trustworthiness of data. If **res* is NULL, this value gives an indication of whether the non-existence of data can be trusted or not.

EXAMPLE

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <validator.h>

    int main(int argc, char *argv[])
    {
             struct val_addrinfo *ainfo = NULL;
             int retval;

             if (argc < 2) {
                     printf("Usage: %s <hostname>\n", argv[0]);
                     exit(1);
             }

             retval = val_getaddrinfo(NULL, argv[1], NULL, NULL, &ainfo);
```

```
            if ((retval == 0) && (ainfo != NULL)) {

                    printf("Validation Status = %d [%s]\n",
                           ainfo->ai_val_status,
                           p_val_status(ainfo->ai_val_status));

                    val_freeaddrinfo(ainfo);
            }

            return 0;
    }
```

SEE ALSO

getaddrinfo(3)

libval(3), val_gethostbyname(3), val_query(3)

3.0.13 val_gethostbyname()

NAME

val_gethostbyname(), *val_gethostbyname2()*, *val_gethostbyname_r()*, *val_gethostbyname2_r()* - get DNSSEC-validated network host entry

SYNOPSIS

```
    #include <validator.h>

    extern int h_errno;
    struct hostent *val_gethostbyname(const val_context_t *ctx,
                                      const char *name,
                                      val_status_t *val_status);

    struct hostent *val_gethostbyname2(const val_context_t *ctx,
                                       const char *name,
                                       int af,
                                       val_status_t *val_status);

    int val_gethostbyname_r(const val_context_t *ctx,
                            const char *name,
                            struct hostent *ret,
                            char *buf,
                            size_t buflen,
                            struct hostent **result,
                            int *h_errnop,
                            val_status_t *val_status);

    int val_gethostbyname2_r(const val_context_t *ctx,
                             const char *name,
                             int af,
                             struct hostent *ret,
                             char *buf,
                             size_t buflen,
                             struct hostent **result,
                             int *h_errnop,
                             val_status_t *val_status);

      struct hostent *val_gethostbyaddr(val_context_t * ctx,
                                        const char *addr,
                                        int len,
                                        int type,
                                        val_status_t * val_status);

      int val_gethostbyaddr_r(val_context_t * ctx,
                              const char *addr,
                              int len,
                              int type,
                              struct hostent *ret,
                              char *buf,
                              int buflen,
                              struct hostent **result,
                              int *h_errnop,
                              val_status_t * val_status);
```

DESCRIPTION

val_gethostbyname(), *val_gethostbyname2()*, *val_gethostbyname_r()*, *val_gethostbyname2_r()*, *val_gethostbyaddr()* and *val_gethostbyaddr_r()* perform DNSSEC validation of DNS queries. They return a network host entry value of type struct *hostent* and are DNSSEC-aware versions of the *gethostbyname(3)*, *gethostbyname2(3)*, *gethostbyname_r()*, *gethostbyname2_r()*, *gethostbyaddr()* and *gethostbyaddr_r()* functions respectively. (See *gethostbyname(3)* for more information on type struct *hostent*).

val_gethostbyname(), *val_gethostbyname_r()*, *val_gethostbyaddr()*, and *val_gethostbyaddr_r()* support only IPv4 addresses. *val_gethostbyname2()* and *val_gethostbyname2_r()* support both IPv4 and IPv6 addresses.

The *val_gethostbyname_r()*, *val_gethostbyname2_r()* and *val_gethostbyaddr_r()* functions are reentrant versions and can be safely used in multi-threaded applications.

The *ctx* parameter specifies the validation context, which can be set to NULL for default values (see *libval(3)* and **dnsval.conf** for more details on validation contexts and validation policy).

val_gethostbyname(), *val_gethostbyname2()* and *val_gethostbyaddr()* set the global *h_errno* variable to return the resolver error code. The reentrant versions *val_gethostbyname_r()*, *val_gethostbyname2_r()* and *val_gethostbyaddr_r()* use the *h_errnop* parameter to return this value. This ensures thread safety, by avoiding the global *h_errno* variable. *h_errnop* must not be NULL. (See the man page for *gethostbyname(3)* for possible values of *h_errno*.)

The *name*, *af*, *ret*, *buf*, *buflen*, and *result* parameters have the same syntax and semantics as the corresponding parameters for the original *gethostbyname*()* and *gethostbyaddr*()* functions. See the manual page for *gethostbyname(3)* for more details about these parameters.

The *val_status* parameter is used to return the validator error code and must not be NULL. *val_istrusted()* and *val_isvalidated()* can be used to determine the trustworthiness of data and *p_val_status()* can be used to display the status value to the user in ASCII format (See *libval(3)* more for information).

RETURN VALUES

The *val_gethostbyname()*, *val_gethostbyname2()*, and *val_gethostbyaddr()* functions return a pointer to a *hostent* structure when they can resolve the given host name (with or without DNSSEC validation), and NULL if data was not available. The memory for the returned value is statically allocated by these two functions. Hence, the caller must not free the memory for the returned value.

The *val_gethostbyname_r()*, *val_gethostbyname2_r()* and *val_gethostbyaddr_r()* functions return 0 when they can resolve the given host name (with or without DNSSEC validation), and a non-zero error-code on failure.

The *val_gethostbyaddr()* and *val_gethostbyaddr_r()* functions return 0 when they can resolve the given host name (with or without DNSSEC validation), and a non-zero error-code on failure.

The *val_status* parameter gives an indication for trustworthiness of data. If the returned *hostent* structure is NULL, this value gives an indication of whether the non-existence of data can be trusted or not.

EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <validator.h>

int main(int argc, char *argv[])
{
    int val_status;
    struct hostent *h = NULL;

    if (argc < 2) {
        printf("Usage: %s <hostname>\n", argv[0]);
        exit(1);
    }

    h = val_gethostbyname(NULL, argv[1], &val_status);
    printf("h_errno = %d [%s]\n", h_errno, hstrerror(h_errno));
    if (h) {
        printf("Validation Status = %d [%s]\n", val_status,
               p_val_status(val_status));
    }

    return 0;
}
```

NOTES

These functions do not currently read the order of lookup from **/etc/hosts.conf**. At present, the default order is set to consult the **/etc/hosts** file first and then query DNS.

The current versions of these functions do not support NIS lookups.

SEE ALSO

gethostbyname(3), gethostbyname2(3), gethostbyname_r(3), gethostbyname2_r(3)

libval(3)

val_getaddrinfo(3), val_query(3)

3.0.14 val_query()

NAME

val_query(), *val_res_query()*, *val_res_search()* - DNSSEC-validated resolution of DNS queries

SYNOPSIS

```
    #include <validator.h>

    int val_query(const val_context_t *ctx,
                  const char *dname,
                  const u_int16_t class,
                  const u_int16_t type,
                  const u_int8_t flags,
                  struct val_response **resp);

    int val_free_response(struct val_response *resp);

    int val_res_query(const val_context_t *ctx,
                      const char *dname,
                      int class,
                      int type,
                      u_char *answer,
                      int anslen,
                      val_status_t *val_status);

    int val_res_search(val_context_t * ctx,
                       const char *dname,
                       int class_h,
                       int type,
                       u_char * answer,
                       int anslen,
                       val_status_t * val_status);
```

DESCRIPTION

The *val_query()* and *val_res_query()* functions perform DNSSEC validation of DNS queries. They are DNSSEC-aware substitutes for *res_query(3)*. *val_res_search()* is a DNSSEC-aware substitute for the *res_search(3)* function.

The *ctx* parameter is the validator context and can be set to NULL for default settings. More information about this field can be found in *libval(3)*.

The *dname* parameter specifies the domain name, *class* specifies the DNS class and *type* specifies the DNS type.

The *val_query()* function returns results in the *resp* linked list which encapsulates the results into the following structure:

```
struct val_response
{
      unsigned char *vr_response;
      int vr_length;
      val_status_t vr_val_status;
      struct val_response *vr_next;
};
```

The *vr_response* and *vr_length* fields are functionally similar to the *answer* and *anslen* parameters in *res_query(3)*. Memory for the *resp* linked list is internally allocated and must be released after a successful invocation of the function using the *val_free_response()* function. Each element in the *resp* linked list will contain an answer corresponding to a single RRSet in the DNS reply.

The validation status is returned in the *vr_val_status* field of the *val_response* structure for that RRSet. *p_val_status()* returns a brief string description of the error code. *val_istrusted()* determines if the status code indicates that the response can be trusted and *val_isvalidated()* determines if the status code indicates that the response was validated. (See *libval(3)* for further information).

The *flags* parameter controls the scope of validation and name resolution, and the output format. Three values, which may be ORed together, are currently defined for this field. The **VAL_QUERY_MERGE_RRSETS** flag is provided for applications that wish to merge all rrsets into a single response returned in the first element of the *resp* array. The response field of this element will have a format similar to the answer returned by *res_query(3)*. The **VAL_QUERY_DONT_VALIDATE** flag causes the validator to disable validation for this query, and the **VAL_QUERY_NO_DLV** flag causes the validator to disable DLV processing for this query. The last flag is only available if the *libval(3)* library has been compiled with DLV support.

val_res_query() is provided as a closer substitute for *res_query(3)*. It calls *val_query()* internally with the **VAL_QUERY_MERGE_RRSETS** flag and returns the answers in the field answer with length of *anslen*.

val_res_search() performs an operation similar to *val_res_query()*. In addition, it uses the search paths specified within the **/etc/resolv.conf** file to create the fully qualified domain name.

The validation status values for *val_res_query()* and *val_res_search()* functions are returned in their respective *val_status* fields.

RETURN VALUES

The *val_query()* function returns 0 on success. It invokes *resolve_n_check()* internally and errors from this function may be returned.

val_res_query() and *val_res_search()* return the number of bytes received on success and -1 on failure.

EXAMPLES

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <strings.h>
#include <arpa/nameser.h>
#include <validator.h>

#define BUFLEN 8096
#define RESPCOUNT 3

int main(int argc, char *argv[])
{
         int retval;
         int i;
         int class = ns_c_in;
         int type = ns_t_a;
         struct val_response *resp, *iter;

         if (argc < 2) {
             printf("Usage: %s <domain-name>\n", argv[0]);
             exit(1);
         }

         retval = val_query(NULL, argv[1], class, type, 0, &resp);

         if (retval == 0) {
             for (iter=resp; iter; iter=iter->vr_next) {
                  printf("Validation Status = %d [%s]\n",
                          iter->vr_val_status,
                          p_val_status(iter->vr_val_status));
                 }
         }

         free_val_response(resp);

         return 0;
}
```

SEE ALSO

res_query(3)

get_context(3), val_getaddrinfo(3), val_gethostbyname(3)

libval(3)

4 Supporting Modules

A number of Perl modules have been developed for DNSSEC-Tools to assist in maintaining DNSSEC-secured domains. These routines manipulate DNSSEC-Tools files, provide GUI interfaces, and manipulate common command options.

These DNSSEC-Tools Perl modules are:

BootStrap.pm	optionally load Perl modules
QWPrimitives.pm	**QWizard** primitives
conf.pm	DNSSEC-Tools configuration file routines
defaults.pm	DNSSEC-Tools defaults routines
dnssectools.pm	general routines for DNSSEC-Tools
keyrec.pm	*keyrec* file manipulation routines
rolllog.pm	DNSSEC-Tools rollover logging routines
rollmgr.pm	**rollerd** interfaces
rollrec.pm	*rollrec* file manipulation routines
timetrans.pm	time/text conversion routines
tooloptions.pm	DNSSEC-Tools common option routines

This section contains man pages describing these modules.

4.0.15 BootStrap.pm

NAME

Net::DNS::SEC::Tools::BootStrap - Optional loading of Perl modules

SYNOPSIS

```
  use Net::DNS::SEC::Tools::BootStrap;

  dnssec_tools_load_mods(PerlModule => 'Additional help/error text');
```

DESCRIPTION

The DNSSEC-Tools package requires a number of Perl modules that are only needed by some of the tools. This module helps determine at run-time, rather than at installation time, if the right tools are available on the system. If any module fails to load, *dnssec_tools_load_mods()* will display an error message and calls *exit()*. The error message describes how to install a module via CPAN.

The arguments to *dnssec_tools_load_mods()* are given in pairs. Each pair is a module to try to load (and import) and a supplemental error message. If the module fails to load, the supplemental error message will be displayed along with the installation-via-CPAN message. If the error message consists of the string "noerror", then no error message will be displayed before the function exits.

CAVEATS

The module will try to import any exported subroutines from the module into the *main* namespace. This means that the **BootStrap.pm** module is likely to not be useful for importing symbols into other modules. Work-arounds for this are:

- import the symbols by hand

```
       dnssec_tools_load_mods(PerlModule => 'Additional help/error text');

       import PerlModule qw(func1 func2);

       func1(arg1, arg2);
```

- call the fully qualified function name

```
       dnssec_tools_load_mods(PerlModule => 'Additional help/error text');

       PerlModule::func1(arg1, arg2);
```

4.0.16 QWPrimitives.pm

NAME

Net::DNS::SEC::Tools::QWPrimitives - QWizard primitives for DNSSEC-Tools

SYNOPSIS

```
  use Net::DNS::SEC::Tools::QWPrimitives;
  use Getopt::Long::GUI;

  GetOptions(  ...,
               ['GUI:nootherargs',1],
               ['GUI:otherprimaries',dnssec_tools_get_qwprimitives()],
               ['GUI:submodules','getzonefiles','getzonenames'],
            );
```

DESCRIPTION

QWizard is a dynamic GUI-construction kit. It displays a series of questions, then retrieves and acts upon the answers. This module provides access to **QWizard** for DNSSEC-Tools software.

In particular, the *dnssec_tools_get_qwprimitives()* returns a set of primary screens for requesting a set of zone files followed by a set of domain names for those zone files. These are then pushed into the *__otherargs qwparam* variable, which is used by **Getopt::GUI::Long** to generate the *ARGV* list.

SEE ALSO

Getopt::GUI::Long(3), Net::DNS(3), QWizard(3)

4.0.17 conf.pm

NAME

Net::DNS::SEC::Tools::conf - DNSSEC-Tools configuration routines.

SYNOPSIS

```
  use Net::DNS::SEC::Tools::conf;

  %dtconf = parseconfig();

  %dtconf = parseconfig("localzone.keyrec");

  cmdcheck(\%options_hashref);

  $prefixdir = getprefixdir();

  $confdir = getconfdir();

  $conffile = getconffile();

  $statedir = getlocalstatedir();

  erraction(ERR_MSG);
  err("unable to open keyrec file",1);
```

DESCRIPTION

The routines in this module perform configuration operations. Some routines access the DNSSEC-Tools configuration file, while others validate the execution environment.

The DNSSEC tools have a configuration file for commonly used values. These values are the defaults for a variety of things, such as encryption algorithm and encryption key length. The **Net::DNS::SEC::Tools::conf** module provides methods for accessing the configuration data in this file.

dnssec-tools.conf is the filename for the DNSSEC tools configuration file. The full path depends on how DNSSEC-Tools was configured; see the DIRECTORIES section for the complete path. The paths required by **conf.pm** are set at DNSSEC-Tools configuration time.

The DNSSEC tools configuration file consists of a set of configuration value entries, with only one entry per line. Each entry has the "keyword value" format. During parsing, the line is broken into tokens, with tokens being separated by spaces and tabs. The first token in a line is taken to be the keyword. All other tokens in that line are concatenated into a single string, with a space separating each token. The untokenized string is added to a hash table, with the keyword as the value's key.

Comments may be included by prefacing them with the '#' or ';' comment characters. These comments can encompass an entire line or may follow a configuration entry. If a comment shares a line with an entry, value tokenization stops just prior to the comment character.

An example configuration file follows:

```
# Sample configuration entries.

algorithm       rsasha1     # Encryption algorithm.
ksk_length      1024        ; KSK key length.
```

Another aspect of DNSSEC-Tools configuration is the error action used by the DNSSEC-Tools Perl modules. The action dictates whether an error condition will only give an error return, print an error message to STDERR, or print an error message and exit. The *erraction()* and *err()* interfaces are used for these operations.

INTERFACES

parseconfig()

This routine reads and parses the system's DNSSEC tools configuration file. The parsed contents are put into a hash table, which is returned to the caller.

parseconfig(conffile)

This routine reads and parses a caller-specified DNSSEC tools configuration file. The parsed contents are put into a hash table, which is returned to the caller. The routine quietly returns if the configuration file does not exist.

cmdcheck(%options_hashref)

This routine ensures that the needed commands are available and executable. If any of the commands either don't exist or aren't executable, then an error message will be given and the process will exit. If all is well, everything will proceed quietly onwards.

The commands keys currently checked are *zonecheck*, *keygen*, and *zonesign*. The pathnames for these commands are found in the given options hash referenced by *%options_hashref*. If the hash doesn't contain an entry for one of those commands, it is not checked.

getconfdir()

This routine returns the name of the DNSSEC-Tools configuration directory.

getconffile()

This routine returns the name of the DNSSEC-Tools configuration file.

getprefixdir()

This routine returns the name of the DNSSEC-Tools prefix directory.

getlocalstatedir()

This routine returns the name of the local state directory.

erraction(error_action)

This interface sets the error action for DNSSEC-Tools Perl modules. The valid actions are:

ERR_SILENT - Do not print an error message, do not exit.

ERR_MSG - Print an error message, do not exit.

ERR_EXIT - Print an error message, exit.

ERR_SILENT is the default action.

The previously set error action is returned.

err("error message",exit_code

The *err()* interface is used by the DNSSEC-Tools Perl modules to report an error and exit, depending on the error action.

The first argument is an error message to print – if the error action allows error messages to be printed.

The second argument is an exit code – if the error action requires that the process exit.

DIRECTORIES

The default directories for this installation are:

prefix - /usr/local

sysconf - /usr/local/etc

DNSSEC-Tools configuration file - /usr/local/etc/dnssec-tools

SEE ALSO

dnssec-tools.conf(5)

4.0.18 defaults.pm

NAME

Net::DNS::SEC::Tools::defaults - DNSSEC-Tools default values.

SYNOPSIS

```
use Net::DNS::SEC::Tools::defaults;

%defs = dnssec_tools_alldefaults();

$defalg = dnssec_tools_default("algorithm");

$cz_path = dnssec_tools_default("zonecheck");

$ksklife = dnssec_tools_default("ksklife");

@default_names = dnssec_tools_defnames();
```

DESCRIPTION

This module maintains a set of default values used by DNSSEC-Tools programs. This allows these defaults to be centralized in a single place and prevents them from being spread around multiple programs.

INTERFACES

dnssec_tools_alldefaults()

This interface returns a copy of all the DNSSEC-Tools defaults in a hash table.

dnssec_tools_default(default)

This interface returns the value of a DNSSEC-Tools default. The interface is passed *default*, which is the name of a default to look up. The value of this default is returned to the caller.

dnssec_tools_defnames()

This interface returns the names of all the DNSSEC-Tools defaults. No default values are returned, but the default names returned by *dnssec_tools_defnames()* may then be passed to *dnssec_tools_default()*.

DEFAULT FIELDS

The following are the defaults defined for DNSSEC-Tools.

admin-email

This default holds the default email address for the DNSSEC-Tools administrator.

algorithm

This default holds the default encryption algorithm.

enddate

This default holds the default zone life, in seconds.

entropy_msg

This default indicates whether or not **zonesigner** should display an entropy message.

keygen

This default holds the path to the key-generation program.

keygen-opts

This default hold a set of options for the key-generation program.

kskcount

This default holds the default number of KSK keys to generate for a zone.

ksklength

This default holds the default length of a KSK key.

ksklife

This default holds the default lifespan of a KSK key. This is only used for determining when to rollover the KSK key. Keys otherwise have no concept of a lifespan. This is measured in seconds.

lifespan-max

This default is the maximum lifespan of a key.

lifespan-min

This default is the minimum lifespan of a key.

random

This default holds the default random number generator device.

rndc

This default is the default path of the BIND **rndc** program.

roll_logfile

This default is the path to **rollerd**'s log file.

roll_loglevel

This default is the default logging level for **rollerd**.

roll_sleeptime

This default holds the default sleep time used by the **rollerd** rollover daemon.

savekeys

This default indicates whether or not keys should be deleted when they are no longer in use.

tanamedconffile

This default specifies the name of the **named** configuration file.

tadnsvalconffile

This default specifies the name of the **dnsval** configuration file.

tasleeptime

This default holds the default value for how long the daemon should sleep.

tacontact

This is merely a placeholder for the contact information. There is no useful default value for this.

tasmtpserver

This is merely a placeholder for the name of the SMTP server. There is no useful default value for this.

usegui

This default indicates whether or not the DNSSEC-Tools GUI should be used for option entry.

viewimage

This default holds the default image viewer.

zonecheck

This default holds the path to the zone-verification program.

zonecheck-opts

This default hold a set of options for the zone-verification program.

zonesign

This default holds the path to the zone-signing program.

zonesign-opts

This default hold a set of options for the zone-signing program.

zskcount

This default holds the default number of ZSK keys to generate for a zone.

zsklength

This default holds the default length of the ZSK key.

zsklife

This default holds the default lifespan of the ZSK key. This is only used for determining when to rollover the ZSK key. Keys otherwise have no concept of a lifespan. This is measured in seconds.

DNSSEC-TOOLS PROGRAM FIELDS

The following are the defaults holding the paths to the DNSSEC-Tools programs.

blinkenlights

This default holds the path to the DNSSEC-Tools **blinkenlights** program.

cleanarch

This default holds the path to the DNSSEC-Tools **cleanarch** program.

cleankrf

This default holds the path to the DNSSEC-Tools **cleankrf** program.

dtconf

This default holds the path to the DNSSEC-Tools **dtconf** program.

dtconfchk

This default holds the path to the DNSSEC-Tools **dtconfchk** program.

dtdefs

This default holds the path to the DNSSEC-Tools **dtdefs** program.

dtinitconf

This default holds the path to the DNSSEC-Tools **dtinitconf** program.

expchk

This default holds the path to the DNSSEC-Tools **expchk** program.

fixkrf

This default holds the path to the DNSSEC-Tools **fixkrf** program.

genkrf

This default holds the path to the DNSSEC-Tools **genkrf** program.

getdnskeys

This default holds the path to the DNSSEC-Tools **getdnskeys** program.

keyarch

This default holds the path to the DNSSEC-Tools **keyarch** program.

krfcheck

This default holds the path to the DNSSEC-Tools **krfcheck** program.

lskrf

This default holds the path to the DNSSEC-Tools **lskrf** program.

lsroll

This default holds the path to the DNSSEC-Tools **lsroll** program.

rollchk

This default holds the path to the DNSSEC-Tools **rollchk** program.

rollctl

This default holds the path to the DNSSEC-Tools **rollctl** program.

rollerd

This default holds the path to the DNSSEC-Tools **rollerd** program.

rollinit

This default holds the path to the DNSSEC-Tools **rollinit** program.

rolllog

This default holds the path to the DNSSEC-Tools **rolllog** program.

rollrec-editor

This default holds the path to the DNSSEC-Tools **rollrec-editor** program.

rollset

This default holds the path to the DNSSEC-Tools **rollset** program.

signset-editor

This default holds the path to the DNSSEC-Tools **signset-editor** program.

tachk

This default holds the path to the DNSSEC-Tools **tachk** program.

timetrans

This default holds the path to the DNSSEC-Tools **timetrans** program.

trustman

This default holds the path to the DNSSEC-Tools **trustman** program.

zonesigner

This default holds the path to the DNSSEC-Tools **zonesigner** program.

4.0.19 dnssectools.pm

NAME

Net::DNS::SEC::Tools::dnssectools - General routines for the DNSSEC-Tools package.

SYNOPSIS

```
  use Net::DNS::SEC::Tools::dnssectools;

  dt_adminmail($subject,$msgbody,$recipient);

  $zspath = dt_cmdpath('zonesigner');

  $ftype = dt_findtype($path);
```

DESCRIPTION

The **dnssectools** module provides a general set of methods for use with DNSSEC-Tools utilities.

INTERFACES

The interfaces to the **dnssectools** module are given below.

dt_adminmail(subject,msgbody,recipient)

This routine emails a message to the administrative user listed in the DNSSEC-Tools configuration file.

dt_adminmail() requires two parameters, both scalars. The *subject* parameter is the subject for the mail message. The *msgbody* parameter is the body of the mail message.

A third parameter, *recipient*, may be given to specify the message's recipient. If this is not given, then the recipient will be taken from the *admin-email* record of the DNSSEC-Tools configuration file.

Return values:

- 1 - the message was created and sent.
- 0 - an invalid recipient was specified.

dt_cmdpath(command)

This routine returns the path to a specified DNSSEC-Tools command. *command* should be the name only, without any leading directories. The command name is checked to ensure that it is a valid DNSEC-Tools command,

Return values:

- The absolute path to the command is returned if the command is valid.
- Null is returned if the command is not valid.

dt_filetype(path)

This routine returns the type of the file named in *path*. The *rollrec* and *keyrec* records contained therein are counted and a type determination is made.

Return values:

keyrec

At least one *keyrec* record was found and no *rollrec* records were found.

rollrec

At least one *rollrec* record was found and no *keyrec* records were found.

mixed

At least one *rollrec* record and at least one *keyrec* record were found. This is most likely an erroneous file.

unknown

No *rollrec* records nor *keyrec* records were found.

nofile

The specified file does not exist.

SEE ALSO

Mail::Send.pm(3), Net::DNS::SEC::Tools::conf.pm(3)

4.0.20 keyrec.pm

NAME

Net::DNS::SEC::Tools::keyrec - DNSSEC-Tools *keyrec* file operations

SYNOPSIS

```
  use Net::DNS::SEC::Tools::keyrec;

  keyrec_creat("localzone.keyrec");
  keyrec_open("localzone.keyrec");
  keyrec_read("localzone.keyrec");

  @krnames = keyrec_names();

  $krec = keyrec_fullrec("example.com");
  %keyhash = %$krec;
  $zname = $keyhash{"algorithm"};

  $val = keyrec_recval("example.com","zonefile");

  $exists = keyrec_exists("example.com");

  keyrec_add("zone","example.com",\%zone_krfields);
  keyrec_add("key","Kexample.com.+005+12345",\%keydata);

  keyrec_del("example.com");
  keyrec_del("Kexample.com.+005+12345");

  keyrec_setval("zone","example.com","zonefile","db.example.com");

  $setname = keyrec_signset_newname("example.com");

  keyrec_signset_new("zone","example-keys");

  keyrec_signset_addkey("example-keys","Kexample.com+005+12345",
                                        "Kexample.com+005+54321");
  keyrec_signset_addkey("example-keys",@keylist);

  keyrec_signset_delkey("example-keys","Kexample.com+005+12345");

  $flag = keyrec_signset_haskey("example-keys","Kexample.com+005+12345");

  keyrec_signset_clear("example-keys","Kexample.com+005+12345");

  @signset = keyrec_signsets();
```

```
keyrec_settime("zone","example.com");
keyrec_settime("set","signing-set-42");
keyrec_settime("key","Kexample.com.+005+76543");

@keyfields = keyrec_keyfields();
@zonefields = keyrec_zonefields();

keyrec_write();
keyrec_saveas("filecopy.krf);
keyrec_close();
keyrec_discard();

$default_krf = keyrec_defkrf();
```

DESCRIPTION

The **Net::DNS::SEC::Tools::keyrec** module manipulates the contents of a DNSSEC-Tools *keyrec* file. *keyrec* files contain data about zones signed by and keys generated by the DNSSEC-Tools programs. Module interfaces exist for looking up *keyrec* records, creating new records, and modifying existing records.

A *keyrec* file is organized in sets of *keyrec* records. Each *keyrec* must be either of *key* type or *zone* type. Key *keyrec*s describe how encryption keys were generated, zone *keyrec*s describe how zones were signed. A *keyrec* consists of a set of keyword/value entries. The following is an example of a key *keyrec*:

```
  key     "Kexample.com.+005+30485"
        zonename        "example.com"
        keyrec_type     "kskcur"
        algorithm       "rsasha1"
        random          "/dev/urandom"
        ksklength       "512"
        ksklife          "15768000"
        keyrec_gensecs  "1101183727"
        keyrec_gendate  "Tue Nov 23 04:22:07 2004"
```

The first step in using this module **must** be to create a new *keyrec* file or open and read an existing one. The *keyrec_creat()* interface creates a *keyrec* file if it does not exist and opens it. The *keyrec_open()* interface opens an existing *keyrec* file. The *keyrec_read()* interface reads the file and parses it into an internal format. The file's records are copied into a hash table (for easy reference by the **keyrec.pm** routines) and in an array (for preserving formatting and comments.)

After the file has been read, the contents are referenced using *keyrec_fullrec()* and *keyrec_recval()*. The *keyrec* contents are modified using *keyrec_add()*, and *keyrec_setval()*. *keyrec_settime()* will update a *keyrec*'s timestamp to the current time. *keyrec*s may be deleted with the *keyrec_del()* interface.

If the *keyrec* file has been modified, it must be explicitly written or the changes are not saved. *keyrec_write()* saves the new contents to disk. *keyrec_saveas()* saves the in-memory *keyrec* contents to the specified file name, without affecting the original file.

keyrec_close() saves the file and close the Perl file handle to the *keyrec* file. If a *keyrec* file is no longer wanted to be open, yet the contents should not be saved, *keyrec_discard()* gets rid of the data, and closes the file handle **without** saving any modified data.

KEYREC INTERFACES

The interfaces to the **keyrec.pm** module are given below.

keyrec_add(keyrec_type,keyrec_name,fields)

This routine adds a new *keyrec* to the *keyrec* file and the internal representation of the file contents. The *keyrec* is added to both the *%keyrecs* hash table and the *keyreclines* array.

keyrec_type specifies the type of the *keyrec* – "key" or "zone". *keyrec_name* is the name of the *keyrec*. *fields* is a reference to a hash table that contains the name/value *keyrec* fields. The keys of the hash table are always converted to lowercase, but the entry values are left as given.

The *ksklength* entry is only added if the value of the *keyrec_type* field is "kskcur".

The *zsklength* entry is only added if the value of the *keyrec_type* field is "zsk", "zskcur", "zskpub", or "zsknew".

Timestamp fields are added at the end of the *keyrec*. For key *keyrec*s, the *keyrec_gensecs* and *keyrec_gendate* timestamp fields are added. For zone *keyrec*s, the *keyrec_signsecs* and *keyrec_signdate* timestamp fields are added.

If a specified field isn't defined for the *keyrec* type, the entry isn't added. This prevents zone *keyrec* data from getting mingled with key *keyrec* data.

A blank line is added after the final line of the new *keyrec*. After adding all new *keyrec* entries, the *keyrec* file is written but is not closed.

Return values are:

0 success

-1 invalid I¡krtype¿

keyrec_close()

This interface saves the internal version of the *keyrec* file (opened with *keyrec_creat()*, *keyrec_open()* or *keyrec_read()*) and closes the file handle.

keyrec_creat(keyrec_file)

This interface creates a *keyrec* file if it does not exist, and truncates the file if it already exists. It leaves the file in the open state.

keyrec_creat() returns 1 if the file was created successfully. It returns 0 if there was an error in creating the file.

keyrec_defkrf()

This routine returns the default *keyrec* filename from the DNSSEC-Tools configuration file.

keyrec_del(keyrec_name)

This routine deletes a *keyrec* from the *keyrec* file and the internal representation of the file contents. The *keyrec* is deleted from both the *%keyrecs* hash table and the *keyreclines* array.

Only the *keyrec* itself is deleted from the file. Any associated comments and blank lines surrounding it are left intact.

Return values are:

0 successful I¡keyrec¿ deletion

-1 invalid I¡krtype¿ (empty string or unknown name)

keyrec_discard()

This routine removes a *keyrec* file from use by a program. The internally stored data are deleted and the *keyrec* file handle is closed. However, modified data are not saved prior to closing the file handle. Thus, modified and new data will be lost.

keyrec_exists(keyrec_name)

keyrec_exists() returns a boolean indicating if a *keyrec* exists that has the specified *keyrec_name.*

keyrec_fullrec(keyrec_name)

keyrec_fullrec() returns a reference to the *keyrec* specified in *keyrec_name.*

keyrec_keyfields()

This routine returns a list of the recognized fields for a key *keyrec.*

keyrec_names()

This routine returns a list of the *keyrec* names from the file.

keyrec_open(keyrec_file)

This interface opens an existing *keyrec* file. It first attempts to open the *keyrec* file for reading and writing. If this fails, then it attempts to open it read-only.

keyrec_open() returns 1 if the file was opened successfully. It returns 0 if the file does not exists or if there was an error in opening the file.

keyrec_read(keyrec_file)

This interface reads the specified *keyrec* file and parses it into a *keyrec* hash table and a file contents array. *keyrec_read()* **must** be called prior to any of the other **keyrec.pm** calls. If another *keyrec* is already open, then it is saved and closed prior to opening the new *keyrec.*

Upon success, *keyrec_read()* returns the number of *keyrec*s read from the file.

Failure return values:

-1 specified I¡keyrec¿ file doesn't exit

-2 unable to open I¡keyrec¿ file

-3 duplicate I¡keyrec¿ names in file

keyrec_recval(keyrec_name,keyrec_field)

This routine returns the value of a specified field in a given *keyrec*. *keyrec_name* is the name of the particular *keyrec* to consult. *keyrec_field* is the field name within that *keyrec*.

For example, the current *keyrec* file contains the following *keyrec*:

```
zone        "example.com"
            zonefile        "db.example.com"
```

The call:

```
keyrec_recval("example.com","zonefile")
```

will return the value "db.example.com".

keyrec_saveas(keyrec_file_copy)

This interface saves the internal version of the *keyrec* file (opened with *keyrec_creat()*, *keyrec_open()* or *keyrec_read()*) to the file named in the *keyrec_file_copy* parameter. The new file's file handle is closed, but the original file and the file handle to the original file are not affected.

keyrec_setval(keyrec_type,keyrec_name,field,value)

Set the value of a *name/field* pair in a specified *keyrec*. The file is **not** written after updating the value. The value is saved in both *%keyrecs* and in *keyreclines*, and the file-modified flag is set.

keyrec_type specifies the type of the *keyrec*. This is only used if a new *keyrec* is being created by this call. *keyrec_name* is the name of the *keyrec* that will be modified. *field* is the *keyrec* field which will be modified. *value* is the new value for the field.

Return values are:

0 if the creation succeeded

-1 invalid type was given

keyrec_settime(keyrec_type,keyrec_name)

Set the timestamp of a specified *keyrec*. The file is **not** written after updating the value. The value is saved in both *%keyrecs* and in *keyreclines*, and the file-modified flag is set. The *keyrec*'s *keyrec_signdate* and *keyrec_signsecs* fields are modified.

keyrec_write()

This interface saves the internal version of the *keyrec* file (opened with *keyrec_creat()*, *keyrec_open()* or *keyrec_read()*). It does not close the file handle. As an efficiency measure, an internal modification flag is checked prior to writing the file. If the program has not modified the contents of the *keyrec* file, it is not rewritten.

keyrec_zonefields()

This routine returns a list of the recognized fields for a zone *keyrec*.

KEYREC SIGNING-SET INTERFACES

Signing Sets are collections of encryption keys, defined by inclusion in a particular "set" *keyrec*. The names of the keys are in the *keyrec*'s *keys* record, which contains the names of the key *keyrec*s. Due to the way key names are handled, the names in a Signing Set must not contain spaces.

The Signing-Set-specific interfaces are given below.

keyrec_signset_newname(zone_name)

keyrec_signset_newname() creates a name for a new Signing Set. The name will be generated by referencing the *lastset* field in the *keyrec* for zone *zone_name*, if the *keyrec* has such a field. The set index number (described below) will be incremented and the *lastset* with the new index number will be returned as the new Signing Set name. If the zone *keyrec* does not have a *lastset* field, then the default set name of *signing-set-0* will be used.

The set index number is the first number found in the *lastset* field. It doesn't matter where in the field it is found, the first number will be considered to be the Signing Set index. The examples below show how this is determined:

lastset Field	**Index**
signing-set-0	0
signing-0-set	0
1-signing-0-set	1
signing-88-set-1	88
signingset4321	4321

keyrec_signset_new(signing_set_name)

keyrec_signset_new() creates the Signing Set named by *signing_set_name*. It returns 1 if the call is successful; 0 if it is not.

keyrec_signset_addkey(signing_set_name,key_list)

keyrec_signset_addkey() adds the keys listed in *key_list* to the Signing Set named by *signing_set_name*. *key_list* may either be an array or a set or arguments to the routine. The *keyrec* is created if it does not already exist. It returns 1 if the call is successful; 0 if it is not.

keyrec_signset_delkey(signing_set_name,key_name)

keyrec_signset_delkey() deletes the key given in *key_name* to the Signing Set named by *signing_set_name*. It returns 1 if the call is successful; 0 if it is not.

keyrec_signset_haskey(signing_set_name,key_name)

keyrec_signset_haskey() returns a flag indicating if the key specified in *key_name* is one of the keys in the Signing Set named by *signing_set_name*. It returns 1 if the signing set has the key; 0 if it does not.

keyrec_signset_clear(keyrec_name)

keyrec_signset_clear() clears the entire signing set from the *keyrec* named by *keyrec_name*. It returns 1 if the call is successful; 0 if it is not.

keyrec_signsets()

keyrec_signsets() returns the names of the signing sets in the *keyrec* file. These names are returned in an array.

KEYREC INTERNAL INTERFACES

The interfaces described in this section are intended for internal use by the **keyrec.pm** module. However, there are situations where external entities may have need of them. Use with caution, as misuse may result in damaged or lost *keyrec* files.

keyrec_init()

This routine initializes the internal *keyrec* data. Pending changes will be lost. An open *keyrec* file handle will remain open, though the data are no longer held internally. A new *keyrec* file must be read in order to use the **keyrec.pm** interfaces again.

keyrec_newkeyrec(kr_name,kr_type)

This interface creates a new *keyrec*. The *keyrec_name* and *keyrec_hash* fields in the *keyrec* are set to the values of the *kr_name* and *kr_type* parameters. *kr_type* must be either “key” or “zone”.

Return values are:

0 if the creation succeeded

-1 if an invalid I¡keyrec¿ type was given

KEYREC DEBUGGING INTERFACES

The following interfaces display information about the currently parsed *keyrec* file. They are intended to be used for debugging and testing, but may be useful at other times.

keyrec_dump_hash()

This routine prints the *keyrec* file as it is stored internally in a hash table. The *keyrec*s are printed in alphabetical order, with the fields alphabetized for each *keyrec*. New *keyrec*s and *keyrec* fields are alphabetized along with current *keyrec*s and fields. Comments from the *keyrec* file are not included with the hash table.

keyrec_dump_array()

This routine prints the *keyrec* file as it is stored internally in an array. The *keyrec*s are printed in the order given in the file, with the fields ordered in the same manner. New *keyrec*s are appended to the end of the array. *keyrec* fields added to existing *keyrec*s are added at the beginning of the *keyrec* entry. Comments and vertical whitespace are preserved as given in the *keyrec* file.

SEE ALSO

Net::DNS::SEC::Tools::conf(5), Net::DNS::SEC::Tools::keyrec(5)

4.0.21 rolllog.pm

NAME

Net::DNS::SEC::Tools::rolllog - DNSSEC-Tools rollover logging interfaces.

SYNOPSIS

```
use Net::DNS::SEC::Tools::rolllog;

@levels = rolllog_levels();

$curlevel = rolllog_level();
$oldlevel = rolllog_level("info");
$oldlevel = rolllog_level(LOG_ERR,1);

$curlogfile = rolllog_file();
$oldlogfile = rolllog_file("-");
$oldlogfile = rolllog_file("/var/log/roll.log",1);

$loglevelstr = rolllog_str(8)
$loglevelstr = rolllog_str("info")

$ret = rolllog_num("info");

rolllog_log(LOG_INFO,"example.com","zone is valid");
```

DESCRIPTION

The **Net::DNS::SEC::Tools::rolllog** module provides logging interfaces for the rollover programs. The logging interfaces allow log messages to be recorded. **rollerd** must be running, as it is responsible for updating the log file.

Each log message is assigned a particular logging level. The valid logging levels are:

Textual Level	Numeric Level	Meaning
tmi	1	The highest level – all log messages are saved.
expire	3	A verbose countdown of zone expiration is given.
info	4	Many informational messages are recorded.
phase	6	Each zone's current rollover phase is given.
err	8	Errors are recorded.
fatal	9	Fatal errors are saved.

Table 14: Logging Levels

The logging levels include all numerically higher levels. For example, if the logging level is set to **phase**, then **err** and **fatal** messages will also be recorded.

LOGGING INTERFACES

rolllog_levels()

This routine returns an array holding the text forms of the user-settable logging levels. The levels are returned in order, from most verbose to least.

rolllog_level(newlevel,useflag)

This routine sets and retrieves the logging level for **rollerd**. The *newlevel* argument specifies the new logging level to be set. *newlevel* may be given in either text or numeric form.

The *useflag* argument is a boolean that indicates whether or not to give a descriptive message and exit if an invalid logging level is given. If *useflag* is true, the message is given and the process exits; if false, -1 is returned.

If given with no arguments, the current logging level is returned. In fact, the current level is always returned unless an error is found. -1 is returned on error.

rolllog_file(newfile,useflag)

This routine sets and retrieves the log file for **rollerd**. The *newfile* argument specifies the new log file to be set. If *newfile* exists, it must be a regular file.

The *useflag* argument is a boolean that indicates whether or not to give a descriptive message if an invalid logging level is given. If *useflag* is true, the message is given and the process exits; if false, no message is given. For any error condition, an empty string is returned.

rolllog_num(loglevel)

This routine translates a text log level (given in *loglevel*) into the associated numeric log level. The numeric log level is returned to the caller.

If *loglevel* is an invalid log level, -1 is returned.

rolllog_str(loglevel)

This routine translates a log level (given in *loglevel*) into the associated text log level. The text log level is returned to the caller.

If *loglevel* is a text string, it is checked to ensure it is a valid log level. Case is irrelevant when checking *loglevel*.

If *loglevel* is numeric, it is must be in the valid range of log levels. *undef* is returned if *loglevel* is invalid.

rolllog_log(level,group,message)

The *rolllog_log()* interface writes a message to the log file. Log messages have this format:

```
    timestamp: group: message
```

The *level* argument is the message's logging level. It will only be written to the log file if the current log level is numerically equal to or less than *level*.

group allows messages to be associated together. It is currently used by **rollerd** to group messages by the zone to which the message applies.

The *message* argument is the log message itself. Trailing newlines are removed.

SEE ALSO

rollctl(1)

rollerd(8)

Net::DNS::SEC::Tools::rollmgr.pm(3)

4.0.22 rollmgr.pm

NAME

Net::DNS::SEC::Tools::rollmgr - Communicate with the DNSSEC-Tools rollover manager.

SYNOPSIS

```
  use Net::DNS::SEC::Tools::rollmgr;

  $dir = rollmgr_dir();

  $idfile = rollmgr_idfile();

  $id = rollmgr_getid();

  rollmgr_dropid();

  rollmgr_rmid();

  rollmgr_cmdint();

  rollmgr_halt();

  rollmgr_channel(1);
  ($cmd,$data) = rollmgr_getcmd();
  $ret = rollmgr_verifycmd($cmd);

  rollmgr_sendcmd(CHANNEL_CLOSE,ROLLCMD_ROLLZONE,"example.com");

  rollmgr_sendcmd(CHANNEL_WAIT,ROLLCMD_ROLLZONE,"example.com");
  ($retcode, $respmsg) = rollmgr_getresp();
```

DESCRIPTION

The **Net::DNS::SEC::Tools::rollmgr** module provides standard, platform-independent methods for a program to communicate with DNSSEC-Tools' **rollerd** rollover manager. There are two interface classes described here: general interfaces and communications interfaces.

GENERAL INTERFACES

The interfaces to the **Net::DNS::SEC::Tools::rollmgr** module are given below.

rollmgr_dir()

This routine returns **rollerd**'s directory.

rollmgr_idfile()

This routine returns **rollerd**'s id file.

rollmgr_getid()

This routine returns **rollerd**'s process id. If a non-zero value is passed as an argument, the id file will be left open and accessible through the PIDFILE file handle. See the WARNINGS section below.

Return Values:

On success, the first portion of the file contents (up to 80 characters) is returned.

-1 is returned if the id file does not exist.

rollmgr_dropid()

This interface ensures that another instance of **rollerd** is not running and then creates a id file for future reference.

Return Values:

1 - the id file was successfully created for this process

0 - another process is already acting as **rollerd**

rollmgr_rmid()

This interface deletes **rollerd**'s id file.

Return Values:

1 - the id file was successfully deleted

0 - no id file exists

-1 - the calling process is not **rollerd**

-2 - unable to delete the id file

rollmgr_cmdint()

This routine informs **rollerd** that a command has been sent via **rollmgr_sendcmd()**.

Return Values:

-1 - an invalid process id was found for **rollerd**

Anything else indicates the number of processes that were signaled.
(This should only ever be 1.)

rollmgr_halt()

This routine informs **rollerd** to shut down.

In the current implementation, the return code from the *kill()* command is returned.

-1 - an invalid process id was found for **rollerd**

Anything else indicates the number of processes that were signaled.
(This should only ever be 1.)

ROLLERD COMMUNICATIONS INTERFACES

rollmgr_channel(serverflag)

This interface sets up a persistent channel for communications with **rollerd**. If *serverflag* is true, then the server's side of the channel is created. If *serverflag* is false, then the client's side of the channel is created.

Currently, the connection may only be made to the localhost. This may be changed to allow remote connections, if this is found to be needed.

rollmgr_getcmd()

rollmgr_getcmd() retrieves a command sent over **rollerd**'s communications channel by a client program. The command and the command's data are sent in each message.

The command and the command's data are returned to the caller.

rollmgr_sendcmd(closeflag,cmd,data)

rollmgr_sendcmd() sends a command to **rollerd**. The command must be one of the commands from the table below. This interface creates a communications channel to **rollerd** and sends the message. The channel is not closed, in case the caller wants to receive a response from **rollerd**.

The available commands and their required data are:

Command	Data	Purpose
ROLLCMD_DISPLAY	1/0	start/stop **rollerd**'s graphical display
ROLLCMD_DSPUB	zone-name	a DS record has been published
ROLLCMD_DSPUBALL	none	DS records published for all zones
		in KSK rollover phase 6
ROLLCMD_ROLLALL	none	force all zones to start ZSK rollover
ROLLCMD_ROLLKSK	zone-name	force a zone to start KSK rollover
ROLLCMD_ROLLREC	rollrec-name	change **rollerd**'s *rollrec* file
ROLLCMD_ROLLZONE	zone-name	force a zone to start ZSK rollover
ROLLCMD_RUNQUEUE	none	**rollerd** runs through its queue
ROLLCMD_SHUTDOWN	none	stop **rollerd**
ROLLCMD_SLEEPTIME	seconds-count	set **rollerd**'s sleep time
ROLLCMD_STATUS	none	get **rollerd**'s status

Table 15: **rollerd** Commands

The data aren't checked for validity by **rollmgr_sendcmd()**; validity checking is a responsibility of **rollerd**.

If the caller does not need a response from **rollerd**, then *closeflag* should be set to **CHANNEL_CLOSE**; if a response is required then *closeflag* should be **CHANNEL_WAIT**. These values are boolean values, and the constants aren't required.

Return Values:

1 is returned on success.

0 is returned if an invalid command is given.

rollmgr_getresp()

After executing a client command sent via **rollmgr_sendcmd()**, **rollerd** will send a response to the client. **rollmgr_getresp()** allows the client to retrieve the response.

A return code and a response string are returned, in that order. Both are specific to the command sent.

rollmgr_verifycmd(cmd)

rollmgr_verifycmd() verifies that *cmd* is a valid command for **rollerd**.

Return Values:

1 is returned for a valid command.

0 is returned for an invalid command.

WARNINGS

1. **rollmgr_getid()** attempts to exclusively lock the id file. Set a timer if this matters to you.

2. **rollmgr_getid()** has a nice little race condition. We should lock the file prior to opening it, but we can't do so without it being open.

SEE ALSO

rollctl(1)

Net::DNS::SEC::Tools::keyrec.pm(3), Net::DNS::SEC::Tools::rolllog.pm(3),
Net::DNS::SEC::Tools::rollrec.pm(3)

rollerd(8)

4.0.23 rollrec.pm

NAME

Net::DNS::SEC::Tools::rollrec - Manipulate a DNSSEC-Tools *rollrec* file.

SYNOPSIS

```
  use Net::DNS::SEC::Tools::rollrec;

  rollrec_lock();
  rollrec_read("localhost.rollrec");

  @rrnames = rollrec_names();

  $flag = rollrec_exists("example.com");

  $rrec = rollrec_fullrec("example.com");
  %rrhash = %$rrec;
  $zname = $rrhash{"maxttl"};

  $val = rollrec_recval("example.com","zonefile");

  rollrec_add("roll","example.com",\%rollfields);
  rollrec_add("skip","example.com",\%rollfields);

  rollrec_del("example.com");

  rollrec_type("example.com","roll");
  rollrec_type("example.com","skip");

  rollrec_setval("example.com","zonefile","db.example.com");

  rollrec_delfield("example.com","directory");

  rollrec_settime("example.com");
  rollrec_settime("example.com",0);

  @rollrecfields = rollrec_fields();

  $default_file = rollrec_default();

  rollrec_write();
  rollrec_close();
  rollrec_discard();

  rollrec_unlock();
```

DESCRIPTION

The **Net::DNS::SEC::Tools::rollrec** module manipulates the contents of a DNSSEC-Tools *rollrec* file. *rollrec* files describe the status of a zone rollover process, as performed by the DNSSEC-Tools programs. Module interfaces exist for looking up *rollrec* records, creating new records, and modifying existing records.

A *rollrec* file is organized in sets of *rollrec* records. *rollrec*s describe the state of a rollover operation. A *rollrec* consists of a set of keyword/value entries. The following is an example of a *rollrec*:

```
    roll "example.com"
        zonefile          "/dnssec-tools/zones/db.example.com"
        keyrec            "/dnssec-tools/keyrec/example.keyrec"
        directory         "/dnssec-tools/dir-example.com"
        kskphase          "0"
        zskphase          "2"
        maxttl            "86400"
        administrator     "bob@bobhost.example.com"
        phasestart        "Wed Mar 09 21:49:22 2005"
        display           "0"
        loglevel          "info"
        rollrec_rollsecs  "1115923362"
        rollrec_rolldate  "Tue Mar 09 19:12:54 2005"
```

The first step in using this module must be to read the *rollrec* file. The *rollrec_read()* interface reads the file and parses it into an internal format. The file's records are copied into a hash table (for easy reference by the **rollrec.pm** routines) and in an array (for preserving formatting and comments.)

After the file has been read, the contents are referenced using *rollrec_fullrec()* and *rollrec_recval()*. The *rollrec_add()*, *rollrec_setval()*, and *rollrec_settime()* interfaces are used to modify the contents of a *rollrec* record.

If the *rollrec* file has been modified, it must be explicitly written or the changes will not saved. *rollrec_write()* saves the new contents to disk. *rollrec_close()* saves the file and close the Perl file handle to the *rollrec* file. If a *rollrec* file is no longer wanted to be open, yet the contents should not be saved, *rollrec_discard()* gets rid of the data closes and the file handle **without** saving any modified data.

ROLLREC LOCKING

This module includes interfaces for synchronizing access to the *rollrec* files. This synchronization is very simple and relies upon locking and unlocking a single lock file for all *rollrec* files.

rollrec locking is not required before using this module, but it is recommended. The expected use of these facilities follows:

```
    rollrec_lock() || die "unable to lock rollrec file\n";
    rollrec_read();
    ... perform other rollrec operations ...
    rollrec_close();
    rollrec_unlock();
```

Synchronization is performed in this manner due to the way the module's functionality is implemented, as well as providing flexibility to users of the module. It also provides a clear delineation in callers' code as to where and when *rollrec* locking is performed.

This synchronization method has the disadvantage of having a single lockfile as a bottleneck to all *rollrec* file access. However, it reduces complexity in the locking interfaces and cuts back on the potential number of required lockfiles.

Using a single synchronization file may not be practical in large installations. If that is found to be the case, then this will be reworked.

ROLLREC INTERFACES

The interfaces to the **rollrec.pm** module are given below.

rollrec_add(rollrec_type,rollrec_name,fields)

This routine adds a new *rollrec* to the *rollrec* file and the internal representation of the file contents. The *rollrec* is added to both the *%rollrecs* hash table and the *rollreclines* array. Entries are only added if they are defined for *rollrec*s.

rollrec_type is the type of the *rollrec*. This must be either "roll" or "skip". *rollrec_name* is the name of the *rollrec*. *fields* is a reference to a hash table that contains the name/value *rollrec* fields. The keys of the hash table are always converted to lowercase, but the entry values are left as given.

Timestamp fields are added at the end of the *rollrec*. These fields have the key values *rollrec_gensecs* and *rollrec_gendate*.

A blank line is added after the final line of the new *rollrec*. The *rollrec* file is not written after *rollrec_add()*, though it is internally marked as having been modified.

rollrec_del(rollrec_name)

This routine deletes a *rollrec* from the *rollrec* file and the internal representation of the file contents. The *rollrec* is deleted from both the *%rollrecs* hash table and the *rollreclines* array.

Only the *rollrec* itself is deleted from the file. Any associated comments and blank lines surrounding it are left intact. The *rollrec* file is not written after *rollrec_del()*, though it is internally marked as having been modified.

Return values are:

0 successful rollrec deletion

-1 unknown name

rollrec_close()

This interface saves the internal version of the *rollrec* file (opened with *rollrec_read()*) and closes the file handle.

rollrec_delfield(rollrec_name,field)

Deletes the given field from the specified *rollrec*. The file is **not** written after updating the value, but the internal file-modified flag is set. The value is saved in both *%rollrecs* and in *rollreclines*.

Return values:

0 - failure (rollrec not found or rollrec does not contain the field)

1 - success

rollrec_discard()

This routine removes a *rollrec* file from use by a program. The internally stored data are deleted and the *rollrec* file handle is closed. However, modified data are not saved prior to closing the file handle. Thus, modified and new data will be lost.

rollrec_exists(rollrec_name)

This routine returns a boolean flag indicating if the *rollrec* named in *rollrec_name* exists.

rollrec_fullrec(rollrec_name)

rollrec_fullrec() returns a reference to the *rollrec* specified in *rollrec_name*.

rollrec_lock()

rollrec_lock() locks the *rollrec* lockfile. An exclusive lock is requested, so the execution will suspend until the lock is available. If the *rollrec* synchronization file does not exist, it will be created. If the process can't create the synchronization file, an error will be returned. Success or failure is returned.

rollrec_names()

This routine returns a list of the *rollrec* names from the file.

rollrec_read(rollrec_file)

This interface reads the specified *rollrec* file and parses it into a *rollrec* hash table and a file contents array. *rollrec_read()* **must** be called prior to any of the other **rollrec.pm** calls. If another *rollrec* is already open, then it is saved and closed prior to opening the new *rollrec*.

rollrec_read() attempts to open the *rollrec* file for reading and writing. If this fails, then it attempts to open the file for reading only.

Upon success, *rollrec_read()* returns the number of *rollrec*s read from the file.

Failure return values:

-1 specified rollrec file doesn't exit

-2 unable to open rollrec file

-3 duplicate rollrec names in file

rollrec_rectype(rollrec_name,rectype)

Set the type of the specified *rollrec* record. The file is **not** written after updating the value, but the internal file-modified flag is set. The value is saved in both *%rollrecs* and in *rollreclines.*

rollrec_name is the name of the *rollrec* that will be modified. *rectype* is the new type of the *rollrec*, which must be either "roll" or "skip".

Return values:

0 - failure (invalid record type or rollrec not found)

1 - success

rollrec_recval(rollrec_name,rollrec_field)

This routine returns the value of a specified field in a given *rollrec. rollrec_name* is the name of the particular *rollrec* to consult. *rollrec_field* is the field name within that *rollrec.*

For example, the current *rollrec* file contains the following *rollrec.*

```
roll        "example.com"
            zonefile        "db.example.com"
```

The call:

```
rollrec_recval("example.com","zonefile")
```

will return the value "db.example.com".

rollrec_settime(rollrec_name,val)

Set the phase-start timestamp in the *rollrec* specified by *rollrec_name* to the current time. If the optional *val* parameter is given and it is zero, then the phase-start timestamp is set to a null value.

The file is **not** written after updating the value.

rollrec_setval(rollrec_name,field,value)

Set the value of a name/field pair in a specified *rollrec.* The file is **not** written after updating the value, but the internal file-modified flag is set. The value is saved in both *%rollrecs* and in *rollreclines.*

rollrec_name is the name of the *rollrec* that will be modified. If the named *rollrec* does not exist, it will be created as a "roll"-type *rollrec. field* is the *rollrec* field which will be modified. *value* is the new value for the field.

rollrec_unlock()

rollrec_unlock() unlocks the *rollrec* synchronization file.

rollrec_write()

This interface saves the internal version of the *rollrec* file (opened with *rollrec_read()*). It does not close the file handle. As an efficiency measure, an internal modification flag is checked prior to writing the file. If the program has not modified the contents of the *rollrec* file, it is not rewritten.

ROLLREC INTERNAL INTERFACES

The interfaces described in this section are intended for internal use by the **rollrec.pm** module. However, there are situations where external entities may have need of them. Use with caution, as misuse may result in damaged or lost *rollrec* files.

rollrec_init()

This routine initializes the internal *rollrec* data. Pending changes will be lost. An open *rollrec* file handle will remain open, though the data are no longer held internally. A new *rollrec* file must be read in order to use the **rollrec.pm** interfaces again.

rollrec_newrec(type,name)

This interface creates a new *rollrec*. The *rollrec_name* field in the *rollrec* is set to the values of the *name* parameter. The *type* parameter must be either “roll” or “skip”.

rollrec_default()

This routine returns the name of the default *rollrec* file.

ROLLREC DEBUGGING INTERFACES

The following interfaces display information about the currently parsed *rollrec* file. They are intended to be used for debugging and testing, but may be useful at other times.

rollrec_dump_hash()

This routine prints the *rollrec* file as it is stored internally in a hash table. The *rollrec*s are printed in alphabetical order, with the fields alphabetized for each *rollrec*. New *rollrec*s and *rollrec* fields are alphabetized along with current *rollrec*s and fields. Comments from the *rollrec* file are not included with the hash table.

rollrec_dump_array()

This routine prints the *rollrec* file as it is stored internally in an array. The *rollrec*s are printed in the order given in the file, with the fields ordered in the same manner. New *rollrec*s are appended to the end of the array. *rollrec* fields added to existing *rollrec*s are added at the beginning of the *rollrec* entry. Comments and vertical whitespace are preserved as given in the *rollrec* file.

SEE ALSO

lsroll(1)

rollchk(8), rollinit(8)

Net::DNS::SEC::Tools::keyrec(3), Net::DNS::SEC::Tools::keyrec(5)

4.0.24 timetrans.pm

NAME

Net::DNS::SEC::Tools::timetrans - Convert an integer seconds count into text units.

SYNOPSIS

```
  use Net::DNS::SEC::Tools::timetrans;

  $timestring = timetrans(86488);

  $timestring = fuzzytimetrans(86488);
```

DESCRIPTION

The *timetrans*() interface in **Net::DNS::SEC::Tools::timetrans** converts an integer seconds count into the equivalent number of weeks, days, hours, and minutes. The time converted is a relative time, **not** an absolute time. The returned time is given in terms of weeks, days, hours, minutes, and seconds, as required to express the seconds count appropriately.

The *fuzzytimetrans*() interface converts an integer seconds count into the equivalent number of weeks **or** days **or** hours **or** minutes. The unit chosen is that which is most natural for the seconds count. One decimal place of precision is included in the result.

INTERFACES

The interfaces to the **Net::DNS::SEC::Tools::timetrans** module are given below.

timetrans()

This routine converts an integer seconds count into the equivalent number of weeks, days, hours, and minutes. This converted seconds count is returned as a text string. The seconds count must be greater than zero or an error will be returned.

Return Values:

If a valid seconds count was given, the count converted into the appropriate text string will be returned.

An empty string is returned if no seconds count was given or if the seconds count is less than one.

fuzzytimetrans()

This routine converts an integer seconds count into the equivalent number of weeks, days, hours, or minutes. This converted seconds count is returned as a text string. The seconds count must be greater than zero or an error will be returned.

Return Values:

If a valid seconds count was given, the count converted into the appropriate text string will be returned.

An empty string is returned if no seconds count was given or if the seconds count is less than one.

EXAMPLES

timetrans(400) returns 6 minutes, 40 seconds

timetrans(420) returns 7 minutes

timetrans(888) returns 14 minutes, 48 seconds

timetrans(86400) returns 1 day

timetrans(86488) returns 1 day, 28 seconds

timetrans(715000) returns 1 week, 1 day, 6 hours, 36 minutes, 40 second

timetrans(720000) returns 1 week, 1 day, 8 hours

fuzzytimetrans(400) returns 6.7 minutes

fuzzytimetrans(420) returns 7.0 minutes

fuzzytimetrans(888) returns 14.8 minutes

fuzzytimetrans(86400) returns 1.0 day

fuzzytimetrans(86488) returns 1.0 day

fuzzytimetrans(715000) returns 1.2 weeks

fuzzytimetrans(720000) returns 1.2 weeks

SEE ALSO

timetrans(1)

4.0.25 tooloptions.pm

NAME

Net::DNS::SEC::Tools::tooloptions - DNSSEC-Tools option routines

SYNOPSIS

```
  use Net::DNS::SEC::Tools::tooloptions;

  @specopts = ("propagate+", "waittime=i");

  $optsref = opts_cmdopts(@specopts);
  %options = %$optsref;

  $zoneref = opts_zonekr($keyrec_file,$keyrec_name,@specopts);
  %zone_kr = %$zoneref;

  opts_setcsopts(@specopts);

  opts_createkrf();

  opts_suspend();

  opts_restore();

  opts_drop();

  opts_reset();

  opts_gui();

  opts_nogui();

  $oldaction = opts_onerr(1);
  opts_onerr(0);
```

DESCRIPTION

DNSSEC-Tools supports a set of options common to all the tools in the suite. These options may be set from DNSSEC-Tools defaults, values set in the **dnssec-tools.conf** configuration file, in a *keyrec* file, from command-specific options, from command-line options, or from any combination of the five. In order to enforce a common sequence of option interpretation, all DNSSEC-Tools should use the **tooloptions.pm** routines to initialize their options.

tooloptions.pm routines combine data from the aforementioned option sources into a hash table. The hash table is returned to the caller, which will then use the options as needed.

The command-line options are saved between calls. This allows a command may call **tool-options.pm** routines multiple times and still have the command-line options included in the final hash table. This is useful for examining multiple *keyrec*s in a single command. Inclusion of command-line options may be suspended and restored using the *opts_suspend()* and *opts_restore()* calls. Options may be discarded entirely by calling *opts_drop()*; once dropped, command-line options may never be restored. Suspension, restoration, and dropping of command-line options are only effective after the initial **tooloptions.pm** call.

The options sources are combined in this order:

- DNSSEC-Tools Defaults

  The DNSSEC-Tools defaults, as defined in **conf.pm** are put into a hash table, with the option names as the hash key.

- DNSSEC-Tools Configuration File

  The system-wide DNSSEC-Tools configuration file is read and these option values are added to the option collection. Again, the option names are used as the hash key.

- *keyrec* File

  If a *keyrec* file was specified, then the *keyrec* named by *keyrec_name* will be retrieved. The *keyrec*'s fields are added to the hash table. Any field whose keyword matches an existing hash key will override any existing values.

- Command-Specific Options

  Options specific to the invoking commands may be specified in *specopts*. This array is parsed by *Getoptions()* from the **Getopt::Long** Perl module. These options are folded into the hash table; possibly overriding existing hash values. The options given in *specopts* must be in the format required by *Getoptions()*.

- Command-Line Options

  The command-line options are parsed using *Getoptions()* from the **Getopt::Long** Perl module. These options are folded into the hash table; again, possibly overriding existing hash values. The options given in *specopts* must be in the format required by *Getoptions()*.

A reference to the hash table created in these steps is returned to the caller.

EXAMPLE

dnssec-tools.conf has these entries:

```
    ksklength       2048
    zsklength       512
```

example.keyrec has this entry:

```
    key         "Kexample.com.+005+12345"
            zsklength       "1024"
```

zonesigner is executed with this command line:

```
    zonesigner -zsklength 4096 -wait 3600 ...  example.com
```

opts_zonekr("example.keyrec","Kexample.com.+005+12345",("wait=i")) will read each option source in turn, ending up with:

```
    I<ksklength>            512
    I<zsklength>           4096
    I<wait>                 600
```

TOOLOPTIONS INTERFACES

opts_cmdopts(csopts)

This *opts_cmdopts()* call builds an option hash from the system configuration file, a *keyrec*, and a set of command-specific options. A reference to this option hash is returned to the caller.

If *$keyrec_file* is given as an empty string, then no *keyrec* file will be consulted. In this case, it is assumed that *$keyrec_name* will be left out altogether.

If a non-existent *$keyrec_file* is given and *opts_createkrf()* has been called, then the named *keyrec* file will be created. *opts_createkrf()* must be called for each *keyrec* file that must be created, as the **tooloptions** *keyrec*-creation state is reset after *tooloptions()* has completed.

opts_zonekr(keyrec_file,keyrec_name,csopts)

This routine returns a reference to options gathered from the basic option sources and from the zone *keyrec* named by *$keyrec_name*, which is found in *$keyrec_file*. The *keyrec* fields from the zone's KSK and ZSK are folded in as well, but the key's *keyrec_* fields are excluded. This call ensures that the named *keyrec* is a zone *keyrec*; if it isn't, *undef* is returned.

The *$keyrec_file* argument specifies a *keyrec* file that will be consulted. The *keyrec* named by the *$keyrec_name* argument will be loaded. If a *keyrec* file is found and *opts_createkrf()* has been previously called, then the *keyrec* file will be created if it doesn't exist.

If *$keyrec_file* is given as "", then the command-line options are searched for a *-krfile* option. If *$keyrec_name* is given as "", then the name is taken from *$ARGV[0]*.

The *specopts* array contains command-specific arguments; the arguments must be in the format prescribed by the **Getopt::Long** Perl module.

opts_setcsopts(csopts)

This routine saves a copy of the command-specific options given in *csopts*. This collection of options is added to the *csopts* array that may be passed to **tooloptions.pm** routines.

opts_createkrf()

Force creation of an empty *keyrec* file if the specified file does not exist. This may happen on calls to *opts_zonekr()*.

opts_suspend()

Suspend inclusion of the command-line options in building the final hash table of responses.

opts_restore()

Restore inclusion of the command-line options in building the final hash table of responses.

opts_drop()

Discard the command-line options. They will no longer be available for inclusion in building the final hash table of responses for this execution of the command.

opts_reset()

Reset an internal flag so that the command-line arguments may be re-examined. This is usually only useful if the arguments have been modified by the calling program itself.

opts_gui()

Set an internal flag so that command arguments may be specified with a GUI. GUI usage requires that **Getopt::Long::GUI** is available. If it isn't, then **Getopt::Long** will be used.

opts_nogui()

Set an internal flag so that the GUI will not be used for specifying command arguments.

opts_onerr(exitflag)

Set an internal flag indicating what should happen if an invalid option is specified on the command line. If *exitflag* is non-zero, then the process will exit on an invalid option; if it is zero, then the process will not exit. The default action is to report an error without exiting.

The old exit action is returned.

SEE ALSO

zonesigner(8)

Getopt::Long(3)

Net::DNS::SEC::Tools::conf(3), Net::DNS::SEC::Tools::defaults(3),
Net::DNS::SEC::Tools::keyrec(3)

Net::DNS::SEC::Tools::keyrec(5)

5 Data Files

Several data files are used by the DNSSEC-Tools components.

These DNSSEC-Tools files are:

dnssec-tools.conf	Configuration file for DNSSEC-Tools programs
dnsval.conf	Configuration policy for the DNSSEC-Tools validator library
keyrec files	Key and zone configuration files for DNSSEC-Tools programs
rollrec files	Rollover configuration files for DNSSEC-Tools programs
donuts rules	Rule definition files for **donuts**.
blinkenlights.conf	Configuration file for **blinkenlights**.

This section contains man pages describing these commands.

5.0.26 dnssec-tools.conf

NAME

dnssec-tools.conf - Configuration file for the DNSSEC-Tools programs

DESCRIPTION

This file contains configuration information for the DNSSEC-Tools programs. These configuration data are used if nothing else has been specified for a particular program. The **conf.pm** module is used to parse this configuration file.

The recognized configuration fields are described in the Configuration Records section below. Some configuration entries are optional and a configuration file need not contain a complete list of entries.

A line in the configuration file contains either a comment or a configuration entry. Comment lines start with either a '#' character or a ';' character. Comment lines and blank lines are ignored by the DNSSEC-Tools programs.

Configuration entries are in a *keyword/value* format. The keyword is a character string that contains no whitespace. The value is a tokenized list of the remaining character groups, with each token separated by a single space.

True/false flags must be given a **1** (true) or **0** (false) value.

Configuration Records

The following records are recognized by the DNSSEC-Tools programs. Not every DNSSEC-Tools program requires each of these records.

admin-email

The email address for the DNSSEC-Tools administrator.

algorithm

The default encryption algorithm to be passed to **dnssec-keygen**.

archivedir

The pathname to the archived-key directory.

default_keyrec

The default *keyrec* filename to be used by the **keyrec.pm** module.

endtime

The zone default expiration time to be passed to **dnssec-signzone**.

entropy_msg

A true/false flag indicating if the **zonesigner** command should display a message about entropy generation. This is primarily dependent on the implementation of a system's random number generation.

keyarch

The path to the DNSSEC-Tools **keyarch** command.

keygen

The path to the **dnssec-keygen** command.

keygen-opts

Options to pass to the **dnssec-keygen** command.

kskcount

The default number of KSK keys that will be generated for each zone.

ksklength

The default KSK key length to be passed to **dnssec-keygen**.

ksklife

The default length of time between KSK roll-overs. This is measured in seconds.

This value is **only** used for key roll-over. Keys do not have a life-time in any other sense.

lifespan-max

The maximum length of time a key should be in use before it is rolled over. This is measured in seconds.

lifespan-min

The minimum length of time a key should be in use before it is rolled over. This is measured in seconds.

random

The random device generator to be passed to **dnssec-keygen**.

roll_logfile

The log file used by **rollerd**.

roll_loglevel

The default logging level used by **rollerd**. The valid levels are defined and described in **rollmgr.pm**.

roll_sleeptime

The number of seconds **rollerd** must wait at the end of each zone-checking cycle.

savekeys

A true/false flag indicating if old keys should be moved to the archive directory.

usegui

Flag to allow/disallow usage of the GUI for specifying command options.

zonecheck

The path to the **named-checkzone** command.

zonecheck-opts

Options to pass to the **named-checkzone** command.

zonesign

The path to the **dnssec-signzone** command.

zonesign-opts

Options to pass to the **dnssec-signzone** command.

zonesigner

The path to the DNSSEC-Tools **zonesigner** command.

zskcount

The default number of ZSK keys that will be generated for each zone.

zsklength

The default ZSK key length to be passed to **dnssec-keygen**.

zsklife

The default length of time between ZSK roll-overs. This is measured in seconds.

This value is **only** used for key roll-over. Keys do not have a life-time in any other sense.

Sample Times

Several configuration fields measure various times. This section is a convenient reference for several common times, as measured in seconds.

```
3600        - hour
86400       - day
604800      - week
2592000     - 30-day month
15768000    - half-year
31536000    - year
```

Example File

The following is an example **dnssec-tools.conf** configuration file.

```
#
# Settings for DNSSEC-Tools administration.
#
admin-email     tewok@squirrelking.net
```

```
#
# Paths to required programs.  These may need adjusting for
# individual hosts.
#
keygen          /usr/local/sbin/dnssec-keygen
rndc            /usr/local/sbin/rndc
viewimage       /usr/X11R6/bin/xview
zonecheck       /usr/local/sbin/named-checkzone
zonecheck-opts  -k ignore
zonesign        /usr/local/sbin/dnssec-signzone

keyarch         /usr/bin/keyarch
rollrec-chk     /usr/bin/rollrec-check
zonesigner      /usr/bin/zonesigner

#
# Settings for dnssec-keygen.
#
algorithm     rsasha1
ksklength     2048
zsklength     1024
random        /dev/urandom


#
# Settings for dnssec-signzone.
#
endtime         +2592000            # RRSIGs good for 30 days.

#
# Life-times for keys.  These defaults indicate how long a key has
# between roll-overs.  The values are measured in seconds.
#
ksklife         15768000            # Half-year.
zsklife         604800              # One week.
lifespan-max    94608000            # Two years.
lifespan-min    3600                # One hour.

#
# Settings that will be noticed by zonesigner.
#
archivedir          /usr/local/etc/dnssec-tools/KEY-SAFE
default_keyrec      default.krf
entropy_msg         0
savekeys            1
zskcount            1
```

```
#
# Settings for rollover-manager.
#
roll_logfile    /usr/local/etc/dnssec-tools/log-rollerd
roll_loglevel   info
roll_sleeptime  60

#
# GUI-usage flag.
#
usegui                0
```

SEE ALSO

dtinitconf(8), dtconfchk(8), keyarch(8), rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::conf.pm(3),
Net::DNS::SEC::Tools::keyrec.pm(3),
Net::DNS::SEC::Tools::rollmgr.pm(3)

5.0.27 dnsval.conf

NAME

/etc/dnsval.conf, **/etc/resolv.conf**, **/etc/root.hints** - Configuration policy for the DNSSEC validator library *libval(3)*
val_add_valpolicy() - Dynamically add a new policy to the validator context

val_remove_valpolicy() - Remove a dynamically added policy from the validator context

SYNOPSIS

```
    int val_add_valpolicy(val_context_t *context,
  const char *keyword,
                        char *zone,
                        char *value,
                        long ttl,
                        val_policy_entry_t **pol);

    int val_remove_valpolicy(val_context_t *context,
     val_policy_entry_t *pol);
```

DESCRIPTION

Applications can use local policy to influence the validation outcome. Examples of local policy elements include trust anchors for different zones and untrusted algorithms for cryptographic keys and hashes. Local policy may vary for different applications and operating scenarios.

The *val_add_valpolicy()* function can be used to dynamically add a new policy for a given context. The *keyword*, *zone* and *value* arguments are identical to *KEYWORD*, *zone* and *additional-data* defined below for **/etc/dnsval.conf**. *ttl* specifies the duration in seconds for which the policy is kept in effect. A value of **-1** adds to policy to the context indefinitely. A handle to the newly added policy is returned in **pol*. This structure is opaque to the applications; applications must not modify the contents of the memory returned in **pol*.

Applications may also revoke the effects of a newly added policy, *pol*, before the expiry of its timeout interval using the *val_remove_valpolicy()* policy.

The validator library reads configuration information from three files: **/etc/resolv.conf**, **/etc/root.hints**, and **/etc/dnsval.conf**.

/etc/resolv.conf

The *nameserver* and *search* options are supported in the **resolv.conf** file.

This *nameserver* option is used to specify the IP address of the name server to which queries must be sent by default. For example,

```
nameserver 10.0.0.1
```

This *search* option is used to specify the search path for issuing queries. For example,

```
search test.dnssec-tools.org dnssec-tools.org
```

If the **/etc/resolv.conf** file contains no name servers, the validator tries to recursively answer the query using information present in **/etc/root.hints**.

/etc/root.hints

The **/etc/root.hints** file contains bootstrapping information for the resolver while it attempts to recursively answer queries. The contents of this file may be generated by the following command:

```
dig @e.root-servers.net . ns > root.hints
```

/etc/dnsval.conf

The **/etc/dnsval.conf** file contains the validator policy. It consists of a sequence of the following "policy-fragments":

```
<label> <KEYWORD> <zone> <additional-data> [<zone> <additional-data> ];
```

Policies are identified by simple text strings called labels, which must be unique within the configuration system. For example, "browser" could be used as the label that defines the validator policy for all web-browsers in a system. A label value of ":" identifies the default policy, the policy that is used when a NULL context is specified as the *ctx* parameter for interfaces listed in *libval(3)*, *val_getaddrinfo(3)*, and *val_gethostbyname(3)*. The default policy is unique within the configuration system.

KEYWORD is the specific policy component that is specified within the policy fragment. The format of *additional-data* depends on the keyword specified.

If multiple policy fragments are defined for the same label and keyword combination then the last definition in the file is used.

The following keywords are defined for **dnsval.conf**:

trust-anchor

Specifies the trust anchors for a sequence of zones. The additional data portion for this keyword is a quoted string containing the RDATA portion for the trust anchor's DNSKEY.

zone-security-expectation

Specifies the local security expectation for a zone. The additional data portion for this keyword is the zone's trust status - ignore, validate, trusted, or untrusted. The default zone security expectation is validate.

provably-unsecure-status

Specifies if the provably unsecure condition must be considered as trusted or not. The additional data portion for this keyword is the trust status for the provably unsecure condition for a given zone - trusted, or untrusted. The default provably unsecure status is trusted.

clock-skew

Specifies how many seconds of clock skew is acceptable when verifying signatures for data from a given zone. The additional data portion for this keyword is the number of seconds of clock skew that is acceptable. A value of -1 completely ignores inception and expiration times on signatures for data from a given zone. The default clock skew is 0.

nsec3-max-iter [only if LIBVAL_NSEC3 is enabled]

Specifies the maximum number of iterations allowable while computing a zone's NSEC3 hash. A value of -1 does not place a maximum limit on the number of iterations. This is also the default setting for a zone.

dlv-trust-points [only if LIBVAL_DLV is enabled]

Specifies the DLV tree for the target zone.

EXAMPLE

The **/etc/dnsval.conf** configuration file might appear as follows:

: trust-anchor

dnssec-tools.org.

"257 3 5 AQO8XS4y9r77X9SHBmrxMoJf1Pf9AT9Mr/L5BBGtO9/e9f/zl4FFgM2l B6M2XEm6mp6mit4tzpB/sAEQw1McYz6bJdKkTiqtuWTCfDmgQhI6/Ha0 EfGPNSqnY 99FmbSeWNIRaa4fgSCVFhvbrYq1nXkNVyQPeEVHkoDNCAlr qOA3lw=="

netsec.tislabs.com.

"257 3 5 AQO8XS4y9r77X9SHBmrxMoJf1Pf9AT9Mr/L5BBGtO9/e9f/zl4FFgM2l B6M2XEm6mp6mit4tzpB/sAEQw1McYz6bJdKkTiqtuWTCfDmgQhI6/Ha0 EfGPNSqnY 99FmbSeWNIRaa4fgSCVFhvbrYq1nXkNVyQPeEVHkoDNCAlr qOA3lw=="

;

browser zone-security-expectation

org ignore

net ignore

dnssec-tools.org validate

com ignore

;

: provably-unsecure-status

. trusted

net untrusted

```
        ;

mta clock-skew

        . 0
        fruits.netsec.tislabs.com. -1
        ;

: nsec3-max-iter

        . 30
        ;

browser dlv-trust-points

        . dlv.isc.org
        ;
```

FILES

/etc/resolv.conf, **/etc/root.hints**, **/etc/dnsval.conf**

SEE ALSO

libval(3)

5.0.28 Keyrec Files

NAME

keyrec - Zone and key data used by DNSSEC-Tools programs.

DESCRIPTION

keyrec files contain data about zones signed by and keys generated by the DNSSEC-Tools programs. A *keyrec* file is organized in sets of *keyrec* records. Each *keyrec* record must be either of *zone* type, *set* type, or *key* type. Zone *keyrec*s describe how the zones were signed. Set *keyrec*s describe sets of key *keyrec*s. Key *keyrec*s describe how encryption keys were generated. A *keyrec* consists of a set of keyword/value entries.

The DNSSEC-Tools **keyrec.pm** module manipulates the contents of a *keyrec* file. Module interfaces exist for looking up *keyrec* records, creating new records, and modifying existing records.

Comment lines and blank lines are ignored by the DNSSEC-Tools programs. Comment lines start with either a '#' character or a ';' character.

A *keyrec*'s name may consist of alphabetic characters, numbers, and several special characters. The special characters are the minus sign, the plus sign, the underscore, the comma, the period, and the colon.

The values in a *keyrec*'s entries may consist of alphabetic characters, numbers, and several special characters. The special characters are the minus sign, the plus sign, the underscore, the comma, the period, the colon, the forward-slash, the space, and the tab.

FIELDS

The fields in a *keyrec* record are described in this section. The fields in each type of record (zone, set, key) are described in their own subsection.

Zone Keyrec Fields

archivedir

The name of the key archive directory for this zone.

endtime

The time when the zone's SIG records expire. This field is passed to **dnssec-signzone** as the argument to the *-e* option.

gends

Boolean value to indicate whether or not DS records should be generated for the zone.

keyrec_signdate

The textual timestamp of the zone *keyrec*'s last update. This is a translation of the *keyrec_signsecs* field.

keyrec_signsecs

The numeric timestamp of the zone *keyrec*'s last update. This is measured in seconds since the epoch.

ksdir

The name of the directory to hold the zone's keyset files.

kskcount

The number of KSKs to generate for the zone.

kskcur

The name of the zone's Current KSK signing set. This is used as the name of the signing set of KSK keys *keyrec* fields.

kskpub

The name of the zone's Published KSK signing set.

kskdirectory

The directory that holds the KSK keys.

lastset

The most recently generated signing set for the zone.

serial

The most recent serial number for the zone.

szopts

Optional arguments passed to the **dnssec-signzone** command.

signedzone

The name of the signed zone file for this zone.

zonefile

The name of the zone file for this zone.

zskcount

The number of ZSKs to generate for the zone.

zskcur

The name of the signing set for the current ZSK keys. This is the name of the signing set's set *keyrec*.

zskdirectory

The directory that holds the ZSK keys.

zskpub

The name of the signing set for the current ZSK keys. This is the name of the signing set's set *keyrec*.

zsknew

The name of the signing set for the current ZSK keys. This is the name of the signing set's set *keyrec*.

Set Keyrec Fields

keys

The list of keys in this signing set. Each key listed should have a corresponding key *keyrec* whose name matches the key name.

keyrec_setdate

The textual timestamp of the signing set's last modification. This is a translation of the *keyrec_setsecs* field.

keyrec_setsecs

The numeric timestamp of the signing set's last modification. This is measured in seconds since the epoch.

zonename

The name of the zone for which this signing set was generated.

Key Keyrec Fields

algorithm

The encryption algorithm used to generate this key.

keypath

The path to the key. This may be an absolute or relative path, but it should be one which **zonesigner** may use (in conjunction with other *keyrec* fields to find the key.

keyrec_gendate

The textual timestamp of the key's creation. This is a translation of the *keyrec_gensecs* field.

keyrec_gensecs

The numeric timestamp of the key's creation. This is measured in seconds since the epoch.

kgopts

Additional options to pass to the **dnssec-keygen** command.

ksklength

The length of a KSK key. This is only included in *keyrec*s for KSK keys.

ksklife

The life of a KSK key. This is only included in *keyrec*s for KSK keys.

random

The random number generator used to generate this key.

zonename

The name of the zone for which this key was generated.

zsklength

The length of a ZSK key. This is only included in *keyrec*s for ZSK keys.

zsklife

The life of a ZSK key. This is only included in *keyrec*s for ZSK keys.

EXAMPLES

The following is an example of a zone *keyrec*:

```
zone        "example.com"
        zonefile        "db.example.com"
        signedzone      "db.example.com.signed"
        endtime         "+604800"
        archivedir      "/usr/etc/dnssec-tools/key-vault"
        kskcur          "signing-set-41"
        kskdirectory    "keydir"
        zskcur          "signing-set-42"
        zskpub          "signing-set-43"
        zsknew          "signing-set-44"
        lastset         "signing-set-44"
        keyrec_signsecs "1123771721"
        keyrec_signdate "Thu Aug 11 14:48:41 2005"
```

The following is an example of a set *keyrec*:

```
set        "signing-set-42"
        zonename        "example.com"
        keys            "Kexample.com.+005+88888"
        keyrec_setsecs  "1123771350"
        keyrec_setdate  "Thu Aug 11 14:42:30 2005"
```

The following is an example of a key *keyrec*:

```
    key        "Kexample.com.+005+88888"
            zonename        "example.com"
            keyrec_type     "kskcur"
            algorithm       "rsasha1"
            random          "/dev/urandom"
            keypath         "./Kexample.com.+005+88888.key"
            ksklength       "1024"
            keyrec_gensecs  "1123771354"
            keyrec_gendate  "Thu Aug 11 14:42:34 2005"
```

SEE ALSO

lskrf(1)

dnssec-signzone(8), keyarch(8) signset-editor(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec(3)

5.0.29 Rollrec Files

NAME

rollrec - Rollover-related zone data used by DNSSEC-Tools programs.

DESCRIPTION

rollrec files contain data used by the DNSSEC-Tools to manage key rollover. A *rollrec* file is organized in sets of *rollrec* records. Each *rollrec* record describes the rollover state of a single zone and must be either of *roll* type or *skip* type. Zone *rollrec*s record information about currently rolling zones. Skip *rollrec*s record information about zones that are not being rolled. A *rollrec* consists of a set of keyword/value entries.

The DNSSEC-Tools **rollrec.pm** module manipulates the contents of a *rollrec* file. Module interfaces exist for looking up *rollrec* records, creating new records, and modifying existing records.

Comment lines and blank lines are ignored by the DNSSEC-Tools programs. Comment lines start with either a '#' character or a ';' character.

A *rollrec*'s name may consist of alphabetic characters, numbers, and several special characters. The special characters are the minus sign, the plus sign, the underscore, the comma, the period, the colon, the forward-slash, the space, and the tab.

The values in a *rollrec*'s entries may consist of alphabetic characters, numbers, and several special characters. The special characters are the minus sign, the plus sign, the underscore, the comma, the period, the colon, the forward-slash, the space, and the tab.

FIELDS

The fields in a *rollrec* record are:

administrator

This is the email address for the zone's administrative user. If it is not set, the default from the DNSSEC-Tools configuration file will be used.

directory

This field contains the name of the directory in which **rollerd** will execute for the *rollrec*'s zone. If it is not specified, then the normal **rollerd** execution directory will be used.

display

This boolean field indicates whether or not the zone should be displayed by the **blinkenlights** program.

keyrec

The zone's *keyrec* file.

kskphase

The zone's current KSK rollover phase. A value of zero indicates that the zone is not in rollover, but is in normal operation. A numeric value of 1-7 indicates that the zone is in that phase of KSK rollover.

ksk_rolldate

The time at which the zone's last KSK rollover completed. This is only used to provide a human-readable format of the timestamp. It is derived from the *ksk_rollsecs* field.

ksk_rollsecs

The time at which the zone's last KSK rollover completed. This value is used to derive the *ksk_rolldate* field.

loglevel

The **rollerd** logging level for this zone.

maxttl

The maximum time-to-live for the zone. This is measured in seconds.

phasestart

The time-stamp of the beginning of the zone's current phase.

zonefile

The zone's zone file.

zskphase

The zone's current ZSK rollover phase. A value of zero indicates that the zone is not in rollover, but is in normal operation. A value of 1, 2, 3, 4 indicates that the zone is in that phase of ZSK rollover.

zsk_rolldate

The time at which the zone's last ZSK rollover completed. This is only used to provide a human-readable format of the timestamp. It is derived from the *ksk_rollsecs* field.

zsk_rollsecs

The time at which the zone's last ZSK rollover completed. This value is used to derive the *ksk_rolldate* field.

EXAMPLES

The following is an example of a roll *rollrec* record:

```
roll "example.com"
        zonefile        "example.signed"
        keyrec          "example.krf"
        kskphase        "1"
        zskphase        "0"
```

```
            administrator   "bob@bobbox.example.com"
            loglevel        "info"
            maxttl          "60"
            display         "1"
            ksk_rollsecs    "1172614842"
            ksk_rolldate    "Tue Feb 27 22:20:42 2007"
            zsk_rollsecs    "1172615087"
            zsk_rolldate    "Tue Feb 27 22:24:47 2007"
            phasestart      "Mon Feb 20 12:34:56 2007"
```

The following is an example of a skip *rollrec* record:

```
    skip "test.com"
            zonefile        "test.com.signed"
            keyrec          "test.com.krf"
            kskphase        "0"
            zskphase        "2"
            administrator   "tess@test.com"
            loglevel        "info"
            maxttl          "60"
            display         "1"
            ksk_rollsecs    "1172614800"
            ksk_rolldate    "Tue Feb 27 22:20:00 2007"
            zsk_rollsecs    "1172615070"
            zsk_rolldate    "Tue Feb 27 22:24:30 2007"
            phasestart      "Mon Feb 20 12:34:56 2007"
```

SEE ALSO

lsroll(1),

blinkenlights(8), rollerd(8), zonesigner(8)

Net::DNS::SEC::Tools::keyrec(3), Net::DNS::SEC::Tools::rollrec(3)

keyrec(5)

5.0.30 Rule.pm for donuts

NAME

Net::DNS::SEC::Tools::Donuts::Rule - Define donuts DNS record-checking rules

DESCRIPTION

This class wraps around a rule definition which is used by the **donuts** DNS zone file checker. It stores the data that implements a given rule.

Rules are defined in **donuts** rule configuration files using the following syntax. See the **donuts** manual page for details on where to place those files and how to load them.

RULE FILE FORMAT

Each rule file can contain multiple rules. Each rule is composed of a number of parts. Minimally, it must contain a **name** and a **test** portion. Everything else is optional and/or has defaults associated with it. The rule file format follows this example:

```
name: rulename
class: Warning
<test>
    my ($record) = @_;
    return "problem found"
      if ($record{xxx} != yyy);
</test>
```

Further details about each section can be found below. Besides the tokens below, other rule-specific data can be stored in tokens and each rule is a hash of the above tokens as keys and their associated data. However, there are a few exceptions where special tokens imply special meanings. These special tokens include *test* and *init*. See below for details.

Each rule definition within a file should be separated using a blank line.

Lines beginning with the '#' character will be discarded as a comment.

name

The name of the rule. This is mandatory, as the user may need to refer to names in the future for use with the *-i* flag, specifying behavior in configuration files, and for other uses.

By convention, all names should be specified using capital letters and '_' characters between the words. The leftmost word should give an indication of a global test category, such as "DNSSEC". The better-named the rules, the more power the user will have for selecting certain types of rules via **donuts -i** and other flags.

Example:

```
name: DNSSEC_TEST_SOME_SECURE_FEATURE
```

level

The rule's execution level, as recognized by **donuts**. **donuts** will run only those rules at or above **donuts**' current execution level. The execution level is specified by the *-l* option to **donuts**; if not given, then the default execution level is 5.

The default *level* of every rule is 5.

Generally, more serious problems should receive lower numbers and less serious problems should be placed at a higher number. The maximum value is 9, which is reserved for debugging rules only. 8 is the maximum rule level that user-defined rules should use.

Example:

```
name: DNSSEC_TEST_SOME_SECURE_FEATURE
level: 2
```

class

The *class* code indicates the type of problem associated with the rule. It defaults to "*Error*", and the only other value that should be used is "*Warning*".

This value is displayed to the user. Technically, any value could be specified, but using anything other than the *Error/Warning* convention could break portability in future versions.

Example:

```
name: DNSSEC_TEST_SOME_SECURE_FEATURE
class: Warning
```

ruletype

Rules fall into one of two types (currently): *record* or *name*. *record* rules have their test evaluated for each record in a zone file. *name* rules, on the other hand, get called once per name stored in the database. See the *test* description below for further details on the arguments passed to each rule type.

The default value for this clause is *record*.

Example:

```
name: DNSSEC_TEST_SOME_SECURE_FEATURE
ruletype: record
```

type

Rules that test a particular type of record should specify the *type* field with the type of record it will test. The rule will only be executed for records of that type. This will result in less error checking for the user in the *test* section.

For example, if a rule is testing a particular aspect of an MX record, it should specify MX in this field.

Example:

```
name: DNSSEC_TEST_SOME_SECURE_FEATURE
type: MX
```

init

A block of code to be executed immediately. This is useful for boot-strap code to be performed only at start-up, rather than at every rule-test invocation. For example, "*use MODULE;*" type statements should be used in *init* sections.

init sections are wrapped in an XML-like syntax which specifies the start and end of the *init* section of code.

Example:

```
<init>
    use My::Module;
    $value = calculate();
</init>
```

test

A block of code defining the test for each record or name. The test statement follows the same multi-line code specification described in the *init* clause above. Specifically, all the lines between the <test> and </test> braces are considered part of the test code.

The end result must be a subroutine reference which will be called by the **donuts** program. When the code is evaluated, if it does not begin with "sub {" then a "sub {" prefix and "}" suffix will be automatically added to the code to turn the code-snippet into a Perl subroutine.

If the test fails, it should return an error string which will be displayed for the user. The text will be line-wrapped before display (and thus should be unformatted text.) If the test is checking for multiple problems, a reference to an array of error strings may be returned. A return value of a reference to an empty array also indicates no error.

There are two types of tests (currently), and the code snippet is called with arguments which depend on the *ruletype* clause above. These arguments and calling conventions are as follows:

record tests

These code snippets are expected to test a single **Net::DNS::RR** record.

It is called with two arguments:

- the record which is to be tested
- the rule definition itself.

name tests

These code snippets are expected to test all the records associated with a given name record.

It is called with three arguments:

- a hash reference to all the record types associated with that name (e.g., 'A', 'MX', ...) and each value of the hash will contain an array of all the records for that type. (I.e., more than one entry in the array reference will exist for names containing multiple 'A' records.)
- The rule definition.
- The record name being checked (the name associated with the data from 1) above.)

Examples:

```
# local rule to mandate that each record must have a
# TTL > 60 seconds
name: DNS_TTL_AT_LEAST_60
level: 8
type: record
<test>
    return "TTL too small" if ($_[0]->ttl < 60);
</test>

# local policy to mandate that anything with an A record
# must have an HINFO record too
name: DNS_MX_MUST_HAVE_A
level: 8
type: name
<test>
    return "A records must have an HINFO record too"
        if (exists($_[0]{'A'}) && !exists($_[0]{'HINFO'}));
</test>
```

feature: **NAME**

The feature tag prevents this rule from running unless the **NAME** keyword was specified using the *–features* flag.

desc: **DESCRIPTION**

A short description of what the rule tests that will be printed to the user in help output or in the error summary when **donuts** outputs the results.

help: **TOKEN: TOKEN-HELP**

If the rule is configurable via the user's **.donuts.conf** file, this describes the configuration tokens for the user when they request configuration help via the *-H* or *–help-config* flags. Tokens may be used within rules by accessing them within the rule argument passed to the code (the second argument.)

Example:

1) In the rule file (this is an incomplete definition):

```
name:           SOME_TEST
myconfig:       40
help: myconfig: A special number to configure this test
<test>
    my ($record, $rule) = @_;
    # ... use $rule->{'myconfig'}
</test>
```

2) This allows the user to change the value of myconfig via their **.donuts.conf** file:

```
# change SOME_TEST config...
name:     SOME_TEST
myconfig: 40
```

3) and running donuts -H will show the help line for myconfig.

noindent: 1

nowrap: 1

Normally **donuts** will line-wrap the error summary produced by a rule to enable automatic pretty-printing of error results. Sometimes, however, rules may not want this. The *nowrap* option indicates to **donuts** that the output is pre-formatted but should still be indented to align with the output of the rest of the error text (currently about 15 spaces.) The *noindent* tag, however, indicates that neither wrapping nor indenting should be performed, but that the error should be printed as is.

SEE ALSO

donuts(8)

Net::DNS(3), Net::DNS::RR(3)

5.0.31 blinkenlights.conf

NAME

blinkenlights.conf - Configuration file for the DNSSEC-Tools **blinkenlights** program

DESCRIPTION

This file contains configuration information for the DNSSEC-Tools **blinkenlights** program. These configuration data are used as default values The **conf.pm** module is used to parse this configuration file.

A line in this file contains either a comment or a configuration entry. Comment lines start with either a '#' character or a ';' character. Comment lines and blank lines are ignored by the DNSSEC-Tools programs.

Configuration entries are in a *keyword/value* format. The keyword is a character string that contains no whitespace. The value is a tokenized list of the remaining character groups, with each token separated by a single space.

True/false flags must be given a true or false value. True values are: 1, "yes", "on". False values are: 0, "no", "off".

Configuration Records

The following records are recognized by **blinkenlights**.

colors

Toggle indicating whether or not to use different background colors for **blinkenlights** zone stripes. If on, different colors will be used. If off, the *skipcolor* value will be used.

fontsize

The font size used to display information in the **blinkenlights** window. If this is not specified, the default font size is 18.

modify

Toggle indicating whether or not to allow access to **blinkenlights**' zone-modification commands. These commands are the GUI's front-end to some of **rollerd**'s commands. If on, the commands are enabled. If off, the commands are disabled.

shading

Toggle indicating whether or not to use color shading in **blinkenlights**' status column. If on, shading is enabled. If off, shading is disabled.

showskip

Toggle indicating whether or not to display skipped zones in **blinkenlights**' window. If on, skipped zones are displayed. If off, skipped zones are not displayed.

skipcolor

The background color to use in displaying skipped zones. If this is not specified, the default color is grey.

Example File

The following is an example **blinkenlights.conf** configuration file.

```
#
# DNSSEC-Tools configuration file for blinkenlights
#
#   Recognized values:
#         colors          use different colors for stripes (toggle)
#         fontsize        size of demo output font
#         modify          allow modification commands (toggle)
#         shading         shade the status columns (toggle)
#         showskip        show skipped zones (toggle)
#         skipcolor       color to use for skip records

fontsize        24
modify          no

colors          on
skipcolor       orange
showskip        1
shading         yes
```

SEE ALSO

blinkenlights(8), rollerd(8)

Net::DNS::SEC::Tools::conf.pm(3)