## 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, so a command may call **tooloptions.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:

1. 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.
2. 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.

3. *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.

4. 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()*.

5. 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      1024
```

**example.keyrec** has this entry:

```
key         "Kexample.com.+005+12345"
        zsklength        "2048"
```

**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:

| | |
|---|---|
| *ksklength* | 1024 |
| *zsklength* | 4096 |
| *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 is reading with *keyrec_read()*. To ensure it is properly read, *keyrec_close()* is called first.

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.

If the command line contains the *−dtconfig* option, then *opts_zonekr*() sets that option to be the configuration file. It then parses that file and uses it as the source for configuration file data.

*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::GUI::Long** 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.

## COPYRIGHT



## AUTHOR

Wayne Morrison, tewok AT users DOT sourceforge DOT net

## 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)**