# Chapter 3
# gdc

## 3.1 Name

**gdc** - operational user interface for GateD

## 3.2 Synopsis

**gdc [ -q ] [ -n ] [ -p path ] [ -c** *coresize* **] [ -f** *filesize* **] [ -m** *datasize* **]**
**[ -s** *stacksize* **] [ -t** *seconds* **] command**

## 3.3 Description

**gdc** provides a command-line interface for the operation of the GateD routing daemon. It provides support for:

- starting and stopping the daemon
- delivering signals to manipulate the daemon when it is operating
- maintaining and checking syntax of configuration files
- producing and removing dump and core files

**gdc** can reliably determine GateD's running state and produces a reliable exit status when errors occur, which is useful in shell scripts that manipulate GateD. Commands executed using **gdc** and, optionally, error messages produced by the execution of those commands, are logged via the same **syslogd** facility that GateD itself uses, providing an audit trail of operations performed on the daemon.

If installed as a **setuid root** program, **gdc** will allow non-root users who are members of a trusted group (by default the **gdmaint** group) to manipulate the routing daemon and deny access to others. For audit purposes, the name of the user is logged via **syslogd**, along with an indication of each command executed.

## 3.4 Command-line Options

The command-line options are:

**-n**

**-n** (lowercase) specifies to run without changing the kernel forwarding table. **-n** is useful for testing and when operating GateD as a route server that does no forwarding.

**-p**

**-p** (lowercase) specifies the path to the GateD binary. The default is the **SBINDIR** constraint (determined at compile time), appended with **gated**. Typical values are **/usr/sbin/gated** and **/sbin/gated**.

**-q**

**-q** specifies to run quietly. With this option, informational messages that are normally printed to the standard output are suppressed, and error messages are logged via **syslogd** instead of being printed to the standard error output. This is often convenient when running **gdc** from a shell script.

**-t** *seconds*

**-t** *seconds* specifies the time in seconds that **gdc** will spend waiting for GateD to complete certain operations, in particular at termination and startup. By default, *seconds* is set to 10 seconds.

These additional command-line options may be present, depending on the options used to compile **gdc**:

**-c** *coresize*

**-c** *coresize* sets the maximum size of a core dump that a GateD started with **gdc** will produce. **-c** *coresize* is useful on systems where the default maximum core dump size is too small for GateD to produce a full core dump on errors.

**-f** *filesize*

**-f** *filesize* sets the maximum file size that a GateD started with **gdc** will produce. **-f** *filesize* is useful on systems where the default maximum file dump size is too small for GateD to produce a full state dump when requested.

**-m** *datasize*

**-m** *datasize* sets the maximum size of the data segment of a GateD started with **gdc. -m** *datasize* is useful on systems where the default data segment size is too small for GateD to run.

**-s** *stacksize*

**-s** *stacksize* sets the maximum size of stack of a GateD started with **gdc**. **-s** *stacksize* is useful on systems where the default maximum stack size is too small for GateD to run.

The following commands cause signals to be delivered to GateD for various purposes:

**COREDUMP**

**COREDUMP** sends an abort signal to GateD, causing it to terminate with a core dump.

**dump**

**dump** signals GateD to dump its current state into the dump location, which is either **/var/tmp/gated_dump** or **/usr/tmp/gated_dump**.

**interface**

**interface** signals **gated** to recheck the interface configuration. GateD normally does this periodically, but **interface** can be used to force the daemon to check interface status immediately when changes are known to have occurred. This is useful when GateD is running on an operating system that does not support asynchronous notification of interface state changes (for example, via a routing socket). When an interface change is made by the administrator, this option may then be used to notify GateD of the new state immediately instead of waiting for the next poll.

**KILL**

**KILL** causes GateD to terminate ungracefully. It is more desirable to allow GateD to terminate itself gracefully by using the **term** option.

**reconfig**

**reconfig** signals GateD to reread its configuration file, reconfiguring its current state as appropriate.

**term**

**term** signals GateD to terminate after shutting down all operating routing protocols gracefully. Executing this command a second time should cause GateD to terminate even if some protocols have not yet fully shut down.

**toggletrace**

If GateD is currently tracing to a file, **toggletrace** causes tracing to be suspended and the trace file to be closed. If GateD tracing is currently suspended, **toggletrace** causes the trace file to be reopened and tracing initiated. **toggletrace** is useful for moving trace files.

By default, GateD obtains its configuration from a file normally named **/etc/gated.conf**. The **gdc** program also maintains several other versions of the configuration file, in particular those named:

**/etc/gated.conf+**

**/etc/gated.conf+** is the new configuration file. When **gdc** is requested to install a new configuration file, this file is renamed **/etc/gated.conf**.

**/etc/gated.conf-**

**/etc/gated.conf-** is the old configuration file. When **gdc** is requested to install a new configuration file, the previous **/etc/gated.conf** is renamed to this name.

**/etc/gated.conf--**

**/etc/gated.conf--** is the very old configuration file. **gdc** retains the previous old configuration file under this name.

The following commands perform operations related to configuration files:

**checkconf**

**checkconf** checks **/etc/gated.conf** for syntax errors. To ensure that there are no errors in the configuration that would cause running GateD to terminate on reconfiguration, **checkconf** is usually done after changing the configuration file but before sending a **reconfig** signal to the currently running GateD. When **checkconf** is used, **gdc** issues an informational message indicating whether there were parse errors, and, if so, saves the error output in a file for inspection.

**checknew**

Like **checkconf** except that the new configuration file, **/etc/gated.conf+**, is checked instead.

**newconf**

**newconf** moves the **/etc/gated.conf+** file into place as **/etc/gated.conf**, retaining the older versions of the file as described above. **gdc** will decline to do anything when given this command if the new configuration file doesn't exist.

**backout**

**backout** rotates the configuration files in the newer direction, in effect moving the old configuration file to **/etc/gated.conf**. The command will decline to perform the operation if **/etc/gated.conf-** doesn't exist or is zero length, or if the operation would delete an existing, non-zero length **/etc/gated.conf+** file.

**BACKOUT**

**BACKOUT** performs a **backout** operation even if **/etc/gated.conf+** exists and is of non-zero length.

**modeconf**

**modeconf** sets all configuration files to mode **664**, owner **root**, group **gdmaint**. This allows a trusted non-root user to modify the configuration files.

**createconf**

If **/etc/gated.conf+** does not exist, **createconf** creates a zero length file with the file mode set to **664**, owner **root**, group **gdmaint**. This allows a trusted non-root user to install a new configuration file.

The following commands provide support for starting and stopping GateD and for determining its running state:

**running**

**running** determines whether GateD is currently running. This is done by checking to see whether GateD has a lock on the file containing its PID, whether the PID in the file is sensible, and whether there is a running process with that PID. **running** exits with zero status if GateD is running; non-zero otherwise.

**start**

**start** starts GateD. The command returns an error if GateD is already running. Otherwise **start** executes the GateD binary and waits for up to the delay interval (10 seconds by default, as set with the **-t** option otherwise) until the newly started process obtains a lock on its PID file. A non-zero exit status is returned if an error is detected while executing the binary, or if a lock is not obtained on its PID file within the specified wait time.

**stop**

**stop** stops GateD, gracefully if possible, ungracefully if not. The command returns an error (with non-zero exit status) if GateD is not currently running. Otherwise it sends a terminate signal to GateD and waits for up to the delay interval (10 seconds by default, as specified with the **-t** option otherwise) for the process to exit. Should GateD fail to exit within the delay interval, it is then signaled again with a second terminate signal. Should it fail to exit by the end of the second delay interval, it is signaled for a third time with a kill signal. This should force immediate termination. **stop** terminates with zero exit status when it detects that GateD has terminated, non-zero otherwise.

**restart**

If GateD is running, **restart** terminates it via the same procedure as is used for the **stop** command above. When the previous GateD terminates, or if it was not running prior to command execution, a new GateD process is executed using the procedures described for the **start** command above. A non-zero exit status is returned if any step in this procedure appears to have failed.

The following commands allow the removal of files created by the execution of some of the commands above:

**rmcore**

**rmcore** removes any existing GateD core dump file.

**rmdump**

**rmdump** removes any existing GateD state dump file.

**`rmparse`**

`rmparse` removes the parse error file generated when a **`checkconf`** or **`checknew`** command is executed and syntax errors are encountered in the configuration file being checked.

**`version`**

`version` shows the version information for GateD. GateD cannot already be running at the time **`version`** is executed. No options of **`gdc`** are used with this command.

## 3.5 Related Files

Many of the default filenames listed below contain the string %s, which is replaced by the name with which GateD is invoked. Normally this is **`gated`**, but if invoked as **`gated-test`**, GateD will by default look for **`/etc/gated-test.conf`**. These paths may all be changed at compilation time.

**`/etc/gated`**

**`/etc/gated`** is the location of the GateD binary. Another popular location is **`/usr/local/sbin/gated`**.

**`/etc/gated.conf`**

**`/etc/gated.conf`** is the location of the current GateD configuration file.

**`/etc/gated.conf+`**

**`/etc/gated.conf+`** is the location of the newer configuration file.

**`/etc/gated.conf-`**

**`/etc/gated.conf-`** is the location of an older configuration file.

**`/etc/gated.conf--`**

**`/etc/gated.conf--`** is the location of a much older configuration file.

**`/etc/gated.pid`**

**`/etc/gated.pid`** is the location where GateD stores its PID. The default is **`/etc/%s.pid`**. Another popular location is **`/var/run/%s.pid`**.

**`/var/tmp/gated_dump`**

**`/var/tmp/gated_dump`** is the location of GateD's state dump file. The default is **`/var/tmp/%s_dump`**. Another popular location is **`/usr/tmp/%s_dump`**.

**`/var/tmp/gated_parse`**

**`/var/tmp/gated_parse`** is where config file parse errors go. The default is **`/var/tmp/%s_parse`**. Another popular location is **`/usr/tmp/%s_parse`**.

**`/var/tmp`**

**`/var/tmp`** is where GateD drops its core file. Another popular location is **`/usr/tmp`**. The core file is usually **`core`**, but some systems use **`core.gated`**.

## 3.6 Bugs

Many commands work only when GateD is installed in the system directory with which it was configured.

There is not yet any way to tell **`gdc`** about systems that name their core dump other than **`core`** (**`core.gated`** is a less common possibility).