NAME

gpgconf – Modify .gnupg home directories

SYNOPSIS

gpgconf [*options*] **––list-components**
gpgconf [*options*] **––list-options** *component*
gpgconf [*options*] **––change-options** *component*

DESCRIPTION

The **gpgconf** is a utility to automatically and reasonable safely query and modify configuration files in the '*.gnupg*' home directory. It is designed not to be invoked manually by the user, but automatically by graphical user interfaces (GUI). ([Please note that currently no locking is done, so concurrent access should be avoided. There are some precautions to avoid corruption with concurrent usage, but results may be inconsistent and some changes may get lost. The stateless design makes it difficult to provide more guarantees.])

gpgconf provides access to the configuration of one or more components of the GnuPG system. These components correspond more or less to the programs that exist in the GnuPG framework, like GPG, GPGSM, DirMngr, etc. But this is not a strict one-to-one relationship. Not all configuration options are available through **gpgconf**. **gpgconf** provides a generic and abstract method to access the most important configuration options that can feasibly be controlled via such a mechanism.

gpgconf can be used to gather and change the options available in each component, and can also provide their default values. **gpgconf** will give detailed type information that can be used to restrict the user's input without making an attempt to commit the changes.

gpgconf provides the backend of a configuration editor. The configuration editor would usually be a graphical user interface program that displays the current options, their default values, and allows the user to make changes to the options. These changes can then be made active with **gpgconf** again. Such a program that uses **gpgconf** in this way will be called GUI throughout this section.

COMMANDS

One of the following commands must be given:

--list-components
List all components. This is the default command used if none is specified.

--check-programs
List all available backend programs and test whether they are runnable.

--list-options *component*
List all options of the component *component*.

--change-options *component*
Change the options of the component *component*.

--check-options *component*
Check the options for the component *component*.

--apply-profile *file*
Apply the configuration settings listed in *file* to the configuration files. If *file* has no suffix and no slashes the command first tries to read a file with the suffix **.prf** from the data directory (**gpgconf --list-dirs datadir**) before it reads the file verbatim. A profile is divided into sections using the bracketed component name. Each section then lists the option which shall go into the respective configuration file.

--apply-defaults
Update all configuration files with values taken from the global configuration file (usually *'/etc/gnupg/gpgconf.conf'*).

--list-dirs [*names*]
Lists the directories used by **gpgconf**. One directory is listed per line, and each line consists of a colon-separated list where the first field names the directory type (for example **sysconfdir**) and the second field contains the percent-escaped directory. Although they are not directories, the socket file names used by **gpg-agent** and **dirmngr** are printed as well. Note that the socket file names and the **homedir** lines are the default names and they may be overridden by command line switches. If *names* are given only the directories or file names specified by the list names are printed without any escaping.

--list-config [*filename*]
List the global configuration file in a colon separated format. If *filename* is given, check that file instead.

--check-config [*filename*]
Run a syntax check on the global configuration file. If *filename* is given, check that file instead.

--query-swdb *package_name* [*version_string*]
Returns the current version for *package_name* and if *version_string* is given also an indicator on whether an update is available. The actual file with the software version is automatically downloaded and checked by **dirmngr**. **dirmngr** uses a thresholds to avoid download the file too often and it does this by default only if it can be done via Tor. To force an update of that file this command can be used:

gpg-connect-agent --dirmngr 'loadswdb --force' /bye

--reload [*component*]
Reload all or the given component. This is basically the same as sending a SIGHUP to the component. Components which don't support reloading are ignored. Without *component* or by using "all" for *component* all components which are daemons are reloaded.

--launch [*component*]
If the *component* is not already running, start it. **component** must be a daemon. This is in general not required because the system starts these daemons as needed. However, external software making direct use of **gpg-agent** or **dirmngr** may use this command to ensure that they are started. Using "all" for *component* launches all components which are daemons.

--kill [*component***]**
Kill the given component that runs as a daemon, including **gpg-agent**, **dirmngr**, and **scdaemon**. A **component** which does not run as a daemon will be ignored. Using "all" for *component* kills all components running as daemons. Note that as of now reload and kill have the same effect for **scdaemon**.

--create-socketdir
Create a directory for sockets below /run/user or /var/run/user. This is command is only required if a non default home directory is used and the /run based sockets shall be used. For the default home directory GnUPG creates a directory on the fly.

--remove-socketdir
Remove a directory created with command **--create-socketdir**.

OPTIONS

The following options may be used:

-o *file*
--output *file*
Write output to *file*. Default is to write to stdout.

-v
--verbose
Outputs additional information while running. Specifically, this extends numerical field values by human-readable descriptions.

-q
--quiet Try to be as quiet as possible.

--homedir *dir*
Set the name of the home directory to *dir*. If this option is not used, the home directory defaults to '*~/.gnupg*'. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable '*GNUPGHOME*' or (on Windows systems) by means of the Registry entry *HKCU\Software\GNU\GnuPG:HomeDir*.

On Windows systems it is possible to install GnuPG as a portable application. In this case only this command line option is considered, all other ways to set a home directory are ignored.

To install GnuPG as a portable application under Windows, create an empty file named '*gpgconf.ctl*' in the same directory as the tool '*gpgconf.exe*'. The root of the installation is then that directory; or, if '*gpgconf.exe*' has been installed directly below a directory named '*bin*', its parent directory. You also need to make sure that the following directories exist and are writable: '*ROOT/home*' for the GnuPG home and '*ROOT/var/cache/gnupg*' for internal cache files.

-n
--dry-run
Do not actually change anything. This is currently only implemented for **--change-options** and can be used for testing purposes.

-r
--runtime
Only used together with **--change-options**. If one of the modified options can be changed in a running daemon process, signal the running daemon to ask it to reparse its configuration file after changing.

This means that the changes will take effect at run-time, as far as this is possible. Otherwise, they will take effect at the next start of the respective backend programs.

--status-fd *n*
Write special status strings to the file descriptor *n*. This program returns the status messages SUCCESS or FAILURE which are helpful when the caller uses a double fork approach and can't easily get the return code of the process.

USAGE

The command **--list-components** will list all components that can be configured with **gpgconf**. Usually, one component will correspond to one GnuPG-related program and contain the options of that program's configuration file that can be modified using **gpgconf**. However, this is not necessarily the case. A component might also be a group of selected options from several programs, or contain entirely virtual options that have a special effect rather than changing exactly one option in one configuration file.

A component is a set of configuration options that semantically belong together. Furthermore, several changes to a component can be made in an atomic way with a single operation. The GUI could for example provide a menu with one entry for each component, or a window with one tabulator sheet per component.

The command **--list-components** lists all available components, one per line. The format of each line is:

name:*description*:*pgmname*:

name This field contains a name tag of the component. The name tag is used to specify the component in all communication with **gpgconf**. The name tag is to be used *verbatim*. It is thus not in any escaped format.

description
The *string* in this field contains a human-readable description of the component. It can be displayed to the user of the GUI for informational purposes. It is *percent-escaped* and *localized*.

pgmname
The *string* in this field contains the absolute name of the program's file. It can be used to unambiguously invoke that program. It is *percent-escaped*.

Example:

```
$ gpgconf --list-components
gpg:GPG for OpenPGP:/usr/local/bin/gpg2:
gpg-agent:GPG Agent:/usr/local/bin/gpg-agent:
scdaemon:Smartcard Daemon:/usr/local/bin/scdaemon:
gpgsm:GPG for S/MIME:/usr/local/bin/gpgsm:
dirmngr:Directory Manager:/usr/local/bin/dirmngr:
```

Checking programs

The command **--check-programs** is similar to **--list-components** but works on backend programs and not on components. It runs each program to test whether it is installed and runnable. This also includes a syntax check of all config file options of the program.

The command **--check-programs** lists all available programs, one per line. The format of each line is:

name:*description*:*pgmname*:*avail*:*okay*:*cfgfile*:*line*:*error*:

name This field contains a name tag of the program which is identical to the name of the component. The name tag is to be used *verbatim*. It is thus not in any escaped format. This field may be empty to indicate a continuation of error descriptions for the last name. The description and pgmname fields are then also empty.

description
The *string* in this field contains a human-readable description of the component. It can be displayed to the user of the GUI for informational purposes. It is *percent-escaped* and *localized*.

pgmname
The *string* in this field contains the absolute name of the program's file. It can be used to unambiguously invoke that program. It is *percent-escaped*.

avail The *boolean value* in this field indicates whether the program is installed and runnable.

okay The *boolean value* in this field indicates whether the program's config file is syntactically okay.

cfgfile If an error occurred in the configuration file (as indicated by a false value in the field **okay**), this field has the name of the failing configuration file. It is *percent-escaped*.

line If an error occurred in the configuration file, this field has the line number of the failing statement in the configuration file. It is an *unsigned number*.

error If an error occurred in the configuration file, this field has the error text of the failing statement in the configuration file. It is *percent-escaped* and *localized*.

In the following example the **dirmngr** is not runnable and the configuration file of **scdaemon** is not okay.

```
$ gpgconf --check-programs
gpg:GPG for OpenPGP:/usr/local/bin/gpg2:1:1:
gpg-agent:GPG Agent:/usr/local/bin/gpg-agent:1:1:
scdaemon:Smartcard Daemon:/usr/local/bin/scdaemon:1:0:
gpgsm:GPG for S/MIME:/usr/local/bin/gpgsm:1:1:
dirmngr:Directory Manager:/usr/local/bin/dirmngr:0:0:
```

The command configuration file in the same manner as **--check-programs**, but only for the component *component*.

Listing options

Every component contains one or more options. Options may be gathered into option groups to allow the GUI to give visual hints to the user about which options are related.

The command lists all options (and the groups they belong to) in the component *component*, one per line. *component* must be the string in the field *name* in the output of the **--list-components** command.

There is one line for each option and each group. First come all options that are not in any group. Then comes a line describing a group. Then come all options that belong into each group. Then comes the next group and so on. There does not need to be any group (and in this case the output will stop after the last non-grouped option).

The format of each line is:

name:*flags*:*level*:*description*:*type*:*alt-type*:*argname*:*default*:*argdef*:*value*

name This field contains a name tag for the group or option. The name tag is used to specify the group or option in all communication with **gpgconf**. The name tag is to be used *verbatim*. It is thus not in any escaped format.

flags The flags field contains an *unsigned number*. Its value is the OR-wise combination of the following flag values:

group (1)
If this flag is set, this is a line describing a group and not an option.

The following flag values are only defined for options (that is, if the **group** flag is not used).

optional arg (2)
If this flag is set, the argument is optional. This is never set for *type* **0** (none) options.

list (4) If this flag is set, the option can be given multiple times.

runtime (8)
If this flag is set, the option can be changed at runtime.

default (16)
If this flag is set, a default value is available.

default desc (32)
If this flag is set, a (runtime) default is available. This and the **default** flag are mutually exclusive.

no arg desc (64)
If this flag is set, and the **optional arg** flag is set, then the option has a special meaning if no argument is given.

no change (128)
If this flag is set, **gpgconf** ignores requests to change the value. GUI frontends should grey out this option. Note, that manual changes of the configuration files are still possible.

level This field is defined for options and for groups. It contains an *unsigned number* that specifies the expert level under which this group or option should be displayed. The following expert levels are defined for options (they have analogous meaning for groups):

basic (0)
This option should always be offered to the user.

advanced (1)
This option may be offered to advanced users.

expert (2)
This option should only be offered to expert users.

invisible (3)
This option should normally never be displayed, not even to expert users.

internal (4)
This option is for internal use only. Ignore it.

The level of a group will always be the lowest level of all options it contains.

description
This field is defined for options and groups. The *string* in this field contains a human-readable description of the option or group. It can be displayed to the user of the GUI for informational purposes. It is *percent-escaped* and *localized*.

type This field is only defined for options. It contains an *unsigned number* that specifies the type of the option's argument, if any. The following types are defined:

Basic types:

none (0)
No argument allowed.

string (1)
An *unformatted string*.

int32 (2)
A *signed number*.

uint32 (3)
An *unsigned number*.

Complex types:

pathname (32)
A *string* that describes the pathname of a file. The file does not necessarily need to exist.

ldap server (33)
A *string* that describes an LDAP server in the format:

hostname:*port*:*username*:*password*:*base_dn*

key fingerprint (34)
A *string* with a 40 digit fingerprint specifying a certificate.

pub key (35)
A *string* that describes a certificate by user ID, key ID or fingerprint.

sec key (36)
A *string* that describes a certificate with a key by user ID, key ID or fingerprint.

alias list (37)
A *string* that describes an alias list, like the one used with gpg's group option. The list consists of a key, an equal sign and space separated values.

More types will be added in the future. Please see the *alt-type* field for information on how to cope with unknown types.

alt-type
This field is identical to *type*, except that only the types **0** to **31** are allowed. The GUI is expected to present the user the option in the format specified by *type*. But if the argument type *type* is not supported by the GUI, it can still display the option in the more generic basic type *alt-type*. The GUI must support all the defined basic types to be able to display all options. More basic types may be added in future versions. If the GUI encounters a basic type it doesn't support, it should report an error and abort the operation.

argname
This field is only defined for options with an argument type *type* that is not **0**. In this case it may contain a *percent-escaped* and *localized string* that gives a short name for the argument. The field may also be empty, though, in which case a short name is not known.

default This field is defined only for options for which the **default** or **default desc** flag is set. If the **default** flag is set, its format is that of an *option argument* (see: [Format conventions], for details). If the default value is empty, then no default is known. Otherwise, the value specifies the default value for this option. If the **default desc** flag is set, the field is either empty or contains a description of the effect if the option is not given.

argdef This field is defined only for options for which the **optional arg** flag is set. If the **no arg desc** flag is not set, its format is that of an *option argument* (see: [Format conventions], for details). If the default value is empty, then no default is known. Otherwise, the value specifies the default argument for this option. If the **no arg desc** flag is set, the field is either empty or contains a description of the effect of this option if no argument is given.

value This field is defined only for options. Its format is that of an *option argument*. If it is empty, then the option is not explicitly set in the current configuration, and the default applies (if any). Otherwise, it contains the current value of the option. Note that this field is also meaningful if the option itself does not take a real argument (in this case, it contains the number of times the option appears).

Changing options

The command to change the options of the component *component* to the specified values. *component* must be the string in the field *name* in the output of the **--list-components** command. You have to provide the options that shall be changed in the following format on standard input:

name:*flags*:*new-value*

name This is the name of the option to change. *name* must be the string in the field *name* in the output of the **--list-options** command.

flags The flags field contains an *unsigned number*. Its value is the OR-wise combination of the following flag values:

default (16)
If this flag is set, the option is deleted and the default value is used instead (if applicable).

new-value
The new value for the option. This field is only defined if the **default** flag is not set. The format is that of an *option argument*. If it is empty (or the field is omitted), the default argument is used (only allowed if the argument is optional for this option). Otherwise, the option will be set to the specified value.

The output of the command is the same as that of **--check-options** for the modified configuration file.

Examples:

To set the force option, which is of basic type **none (0)**:

```
$ echo 'force:0:1' | gpgconf --change-options dirmngr
```

To delete the force option:

```
$ echo 'force:16:' | gpgconf --change-options dirmngr
```

The **--runtime** option can influence when the changes take effect.

Listing global options

Sometimes it is useful for applications to look at the global options file '*gpgconf.conf*'. The colon separated listing format is record oriented and uses the first field to identify the record type:

k This describes a key record to start the definition of a new ruleset for a user/group. The format of a key record is:

k:*user*:*group*:

user This is the user field of the key. It is percent escaped. See the definition of the gpg-conf.conf format for details.

group This is the group field of the key. It is percent escaped.

r This describes a rule record. All rule records up to the next key record make up a rule set for that key. The format of a rule record is:

r:::*component*:*option*:*flag*:*value*:

component
This is the component part of a rule. It is a plain string.

option This is the option part of a rule. It is a plain string.

flag This is the flags part of a rule. There may be only one flag per rule but by using the same component and option, several flags may be assigned to an option. It is a plain string.

value This is the optional value for the option. It is a percent escaped string with a single quotation mark to indicate a string. The quotation mark is only required to distinguish between no value specified and an empty string.

Unknown record types should be ignored. Note that there is intentionally no feature to change the global option file through **gpgconf**.

Get and compare software versions.

The GnuPG Project operates a server to query the current versions of software packages related to GnuPG. **gpgconf** can be used to access this online database. To allow for offline operations, this feature works by having **dirmngr** download a file from **https://versions.gnupg.org**, checking the signature of that file and storing the file in the GnuPG home directory. If **gpgconf** is used and **dirmngr** is running, it may ask **dirmngr** to refresh that file before itself uses the file.

The command **--query-swdb** returns information for the given package in a colon delimited format:

name This is the name of the package as requested. Note that "gnupg" is a special name which is replaced by the actual package implementing this version of GnuPG. For this name it is also not required to specify a version because **gpgconf** takes its own version in this case.

iversion
The currently installed version or an empty string. The value is taken from the command line argument but may be provided by gpg if not given.

status The status of the software package according to this table:

- **-** No information available. This is either because no current version has been specified or due to an error.
- **?** The given name is not known in the online database.
- **u** An update of the software is available.
- **c** The installed version of the software is current.
- **n** The installed version is already newer than the released version.

urgency
If the value (the empty string should be considered as zero) is greater than zero an important update is available.

error This returns an **gpg-error** error code to distinguish between various failure modes.

filedate
This gives the date of the file with the version numbers in standard ISO format (**yyyymmddThhmmss**). The date has been extracted by **dirmngr** from the signature of the file.

verified
This gives the date in ISO format the file was downloaded. This value can be used to evaluate the freshness of the information.

version
This returns the version string for the requested software from the file.

reldate This returns the release date in ISO format.

size This returns the size of the package as decimal number of bytes.

hash This returns a hexified SHA-2 hash of the package.

More fields may be added in future to the output.

FILES

/etc/gnupg/gpgconf.conf
If this file exists, it is processed as a global configuration file.
A commented example can be found in the '*examples*' directory of the distribution.

GNUPGHOME/swdb.lst
A file with current software versions. **dirmngr** creates this file on demand from an online resource.

SEE ALSO

gpg(1), **gpgsm**(1), **gpg-agent**(1), **scdaemon**(1), **dirmngr**(1)

The full documentation for this tool is maintained as a Texinfo manual. If GnuPG and the info program are properly installed at your site, the command

```
info gnupg
```

should give you access to the complete manual including a menu structure and an index.