NAME

gpg-agent - Secret key management for GnuPG

SYNOPSIS

gpg-agent [**--homedir** *dir*] [**--options** *file*] [*options*]
gpg-agent [**--homedir** *dir*] [**--options** *file*] [*options*] **--server**
gpg-agent [**--homedir** *dir*] [**--options** *file*] [*options*] **--daemon** [*command_line*]

DESCRIPTION

gpg-agent is a daemon to manage secret (private) keys independently from any protocol. It is used as a backend for **gpg** and **gpgsm** as well as for a couple of other utilities.

The usual way to run the agent is from the **~/.xsession** file:

```
eval $(gpg-agent --daemon)
```

If you don't use an X server, you can also put this into your regular startup file **~/.profile** or **.bash_profile**. It is best not to run multiple instance of the **gpg-agent**, so you should make sure that only one is running: **gpg-agent** uses an environment variable to inform clients about the communication parameters. You can write the content of this environment variable to a file so that you can test for a running agent. Here is an example using Bourne shell syntax:

```
gpg-agent --daemon --enable-ssh-support
--write-env-file ${HOME}/.gpg-agent-info
```

This code should only be run once per user session to initially fire up the agent. In the example the optional support for the included Secure Shell agent is enabled and the information about the agent is written to a file in the HOME directory. Note that by running gpg-agent without arguments you may test whether an agent is already running; however such a test may lead to a race condition, thus it is not suggested.

The second script needs to be run for each interactive session:

```
if [ -f ${HOME}/.gpg-agent-info ]; then
export GPG_AGENT_INFO
export SSH_AUTH_SOCK
fi
```

It reads the data out of the file and exports the variables. If you don't use Secure Shell, you don't need the last two export statements.

You should always add the following lines to your **.bashrc** or whatever initialization file is used for all shell invocations:

```
GPG_TTY=$(tty)
export GPG_TTY
```

It is important that this environment variable always reflects the output of the **tty** command. For W32 systems this option is not required.

Please make sure that a proper pinentry program has been installed under the default filename (which is system dependant) or use the option **pinentry-program** to specify the full name of that program. It is often useful to install a symbolic link from the actual used pinentry (e.g. '*/usr/bin/pinentry-gtk*') to the expected one (e.g. '*/usr/bin/pinentry*').

COMMANDS

Commands are not distinguished from options except for the fact that only one command is allowed.

--version
Print the program version and licensing information. Note that you cannot abbreviate this command.

--help

-h Print a usage message summarizing the most useful command-line options. Note that you cannot abbreviate this command.

--dump-options
Print a list of all available options and commands. Note that you cannot abbreviate this command.

--server
Run in server mode and wait for commands on the **stdin**. The default mode is to create a socket and listen for commands there.

--daemon [*command line*]
Start the gpg-agent as a daemon; that is, detach it from the console and run it in the background. Because **gpg-agent** prints out important information required for further use, a common way of invoking gpg-agent is: **eval $(gpg-agent --daemon)**to setup the environment variables. The option **--write-env-file** is another way commonly used to do this. Yet another way is creating a new process as a child of gpg-agent: **gpg-agent --daemon /bin/sh**. This way you get a new shell with the environment setup properly; if you exit from this shell, gpg-agent terminates as well.

OPTIONS

--options *file*
Reads configuration from *file* instead of from the default per-user configuration file. The default configuration file is named '*gpg-agent.conf*' and expected in the '*.gnupg*' directory directly below the home directory of the user.

--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 W32 systems) by means of the Registry entry *HKCUSoftwareGNUGnuPG:HomeDir*.

-v

--verbose
Outputs additional information while running. You can increase the verbosity by giving several verbose commands to **gpgsm**, such as -vv.

-q

--quiet
Try to be as quiet as possible.

--batch
Don't invoke a pinentry or do any other thing requiring human interaction.

--faked-system-time *epoch*
This option is only useful for testing; it sets the system time back or forth to *epoch* which is the number of seconds elapsed since the year 1970.

--debug-level *level*
Select the debug level for investigating problems. *level* may be a numeric value or a keyword:

none No debugging at all. A value of less than 1 may be used instead of the keyword.

basic Some basic debug messages. A value between 1 and 2 may be used instead of the keyword.

advanced
More verbose debug messages. A value between 3 and 5 may be used instead of the keyword.

expert
Even more detailed messages. A value between 6 and 8 may be used instead of the keyword.

guru All of the debug messages you can get. A value greater than 8 may be used instead of the keyword. The creation of hash tracing files is only enabled if the keyword is used.

How these messages are mapped to the actual debugging flags is not specified and may change with newer releases of this program. They are however carefully selected to best aid in debugging.

--debug *flags*
This option is only useful for debugging and the behaviour may change at any time without notice. FLAGS are bit encoded and may be given in usual C-Syntax. The currently defined bits are:

0(1) X.509 or OpenPGP protocol related data

1(2) values of big number integers

2(4) low level crypto operations

5(32) memory allocation

6(64) caching

7 (128)
show memory statistics.

9 (512)
write hashed data to files named **dbgmd-000***

10 (1024)
trace Assuan protocol

12 (4096)
bypass all certificate validation

--debug-all
Same as **--debug=0xffffffff**

--debug-wait *n*
When running in server mode, wait *n* seconds before entering the actual processing loop and print the pid. This gives time to attach a debugger.

--no-detach
Don't detach the process from the console. This is mainly useful for debugging.

-s

--sh

-c

--csh Format the info output in daemon mode for use with the standard Bourne shell or the C-shell respectively. The default is to guess it based on the environment variable **SHELL** which is correct in almost all cases.

--write-env-file *file*
Often it is required to connect to the agent from a process not being an inferior of **gpg-agent** and thus the environment variable with the socket name is not available. To help setting up those variables in other sessions, this option may be used to write the information into *file*. If *file* is not specified the default name '{*HOME*}*/.gpg-agent-info*' will be used. The format is suitable to be evaluated by a Bourne shell like in this simple example:

```
eval $(cat file)
eval $(cut -d= -f 1 < file | xargs echo export)
```

--no-grab
Tell the pinentry not to grab the keyboard and mouse. This option should in general not be used to avoid X-sniffing attacks.

--log-file *file*
Append all logging output to *file*. This is very helpful in seeing what the agent actually does. If neither a log file nor a log file descriptor has been set on a Windows platform, the Registry entry **HKCUSoftwareGNUGnuPG:DefaultLogFile**, if set, is used to specify the logging output.

--allow-mark-trusted
Allow clients to mark keys as trusted, i.e. put them into the '*trustlist.txt*' file. This is by default not allowed to make it harder for users to inadvertently accept Root-CA keys.

--ignore-cache-for-signing
This option will let **gpg-agent** bypass the passphrase cache for all signing operation. Note that there is also a per-session option to control this behaviour but this command line option takes precedence.

--default-cache-ttl *n*
Set the time a cache entry is valid to *n* seconds. The default is 600 seconds.

--default-cache-ttl-ssh *n*
Set the time a cache entry used for SSH keys is valid to *n* seconds. The default is 1800 seconds.

--max-cache-ttl *n*
Set the maximum time a cache entry is valid to *n* seconds. After this time a cache entry will be expired even if it has been accessed recently or has been set using **gpg-preset-passphrase**. The default is 2 hours (7200 seconds).

--max-cache-ttl-ssh *n*
Set the maximum time a cache entry used for SSH keys is valid to *n* seconds. After this time a cache entry will be expired even if it has been accessed recently or has been set using **gpg-preset-passphrase**. The default is 2 hours (7200 seconds).

--enforce-passphrase-constraints
Enforce the passphrase constraints by not allowing the user to bypass them using the ''Take it anyway'' button.

--min-passphrase-len *n*
Set the minimal length of a passphrase. When entering a new passphrase shorter than this value a warning will be displayed. Defaults to 8.

--min-passphrase-nonalpha *n*
Set the minimal number of digits or special characters required in a passphrase. When entering a new passphrase with less than this number of digits or special characters a warning will be displayed. Defaults to 1.

--check-passphrase-pattern *file*
Check the passphrase against the pattern given in *file*. When entering a new passphrase matching one of these pattern a warning will be displayed. *file* should be an absolute filename. The default is not to use any pattern file.

Security note: It is known that checking a passphrase against a list of pattern or even against a complete dictionary is not very effective to enforce good passphrases. Users will soon figure up ways to bypass such a policy. A better policy is to educate users on good security behavior and optionally to run a passphrase cracker regularly on all users passphrases to catch the very simple ones.

--max-passphrase-days *n*
Ask the user to change the passphrase if *n* days have passed since the last change. With **--enforce-passphrase-constraints** set the user may not bypass this check.

--enable-passphrase-history
This option does nothing yet.

--pinentry-program *filename*
Use program *filename* as the PIN entry. The default is installation dependent.

--pinentry-touch-file *filename*
By default the filename of the socket gpg-agent is listening for requests is passed to Pinentry, so that it can touch that file before exiting (it does this only in curses mode). This option changes the file passed to Pinentry to *filename*. The special name **/dev/null** may be used to completely disable this feature. Note that Pinentry will not create that file, it will only change the modification and access time.

--scdaemon-program *filename*
Use program *filename* as the Smartcard daemon. The default is installation dependent and can be shown with the **gpgconf** command.

--disable-scdaemon
Do not make use of the scdaemon tool. This option has the effect of disabling the ability to do smartcard operations. Note, that enabling this option at runtime does not kill an already forked scdaemon.

--use-standard-socket

--no-use-standard-socket
By enabling this option **gpg-agent** will listen on the socket named '*S.gpg-agent*', located in the home directory, and not create a random socket below a temporary directory. Tools connecting to **gpg-agent** should first try to connect to the socket given in environment variable *GPG_AGENT_INFO* and then fall back to this socket. This option may not be used if the home directory is mounted on a remote file system which does not support special files like fifos or sockets. Note, that**--use-standard-soc ket** is the default on Windows systems. The default may be changed at build time. It is possible to test at

runtime whether the agent has been configured for use with the standard socket by issuing the command **gpg-agent --use-standard-socket-p** which returns success if the standard socket option has been enabled.

--display *string*

--ttyname *string*

--ttytype *string*

--lc-ctype *string*

--lc-messages *string*

--xauthority *string*
These options are used with the server mode to pass localization information.

--keep-tty

--keep-display
Ignore requests to change the current **tty** or X window system's **DISPLAY** variable respectively. This is useful to lock the pinentry to pop up at the **tty** or display you started the agent.

--enable-ssh-support

Enable the OpenSSH Agent protocol.

In this mode of operation, the agent does not only implement the gpg-agent protocol, but also the agent protocol used by OpenSSH (through a separate socket). Consequently, it should be possible to use the gpg-agent as a drop-in replacement for the well known ssh-agent.

SSH Keys, which are to be used through the agent, need to be added to the gpg-agent initially through the ssh-add utility. When a key is added, ssh-add will ask for the password of the provided key file and send the unprotected key material to the agent; this causes the gpg-agent to ask for a passphrase, which is to be used for encrypting the newly received key and storing it in a gpg-agent specific directory.

Once a key has been added to the gpg-agent this way, the gpg-agent will be ready to use the key.

Note: in case the gpg-agent receives a signature request, the user might need to be prompted for a passphrase, which is necessary for decrypting the stored key. Since the ssh-agent protocol does not contain a mechanism for telling the agent on which display/terminal it is running, gpg-agent's ssh-support will use the TTY or X display where gpg-agent has been started. To switch this display to the current one, the following command may be used:

```
gpg-connect-agent updatestartuptty /bye
```

Although all GnuPG components try to start the gpg-agent as needed, this is not possible for the ssh support because ssh does not know about it. Thus if no GnuPG tool which accesses the agent has been run, there is no guarantee that ssh is abale to use gpg-agent for authentication. To fix this you may start gpg-agent if needed using this simple command:

```
gpg-connect-agent /bye
```

Adding the **--verbose** shows the progress of starting the agent.

All the long options may also be given in the configuration file after stripping off the two leading dashes.

EXAMPLES

The usual way to invoke **gpg-agent** is

```
$ eval $(gpg-agent --daemon)
```

An alternative way is by replacing **ssh-agent** with **gpg-agent**. If for example **ssh-agent** is started as part of the Xsession initialization, you may simply replace **ssh-agent** by a script like:

```
#!/bin/sh

exec /usr/local/bin/gpg-agent --enable-ssh-support --daemon
--write-env-file ${HOME}/.gpg-agent-info $@
```

and add something like (for Bourne shells)

```
if [ -f ${HOME}/.gpg-agent-info ]; then
export GPG_AGENT_INFO
export SSH_AUTH_SOCK
fi
```

to your shell initialization file (e.g. '*~/.bashrc*').

FILES

There are a few configuration files needed for the operation of the agent. By default they may all be found in the current home directory (see: [option --homedir]).

gpg-agent.conf

This is the standard configuration file read by **gpg-agent** on startup. It may contain any valid long option; the leading two dashes may not be entered and the option may not be abbreviated. This file is also read after a**SIGHUP** ho wever only a few options will actually have an effect. This default name may be changed on the command line (see: [option --options]). You should backup this file.

trustlist.txt

This is the list of trusted keys. You should backup this file.

Comment lines, indicated by a leading hash mark, as well as empty lines are ignored. To mark a key as trusted you need to enter its fingerprint followed by a space and a capital letter **S**. Colons may optionally be used to separate the bytes of a fingerprint; this allows to cut and paste the fingerprint from a key listing output. If the line is prefixed with a **!** the key is explicitly marked as not trusted.

Here is an example where two keys are marked as ultimately trusted and one as not trusted:

```
# CN=Wurzel ZS 3,O=Intevation GmbH,C=DE
A6935DD34EF3087973C706FC311AA2CCF733765B S

# CN=PCA-1-Verwaltung-02/O=PKI-1-Verwaltung/C=DE
DC:BD:69:25:48:BD:BB:7E:31:6E:BB:80:D3:00:80:35:D4:F8:A6:CD S

# CN=Root-CA/O=Schlapphuete/L=Pullach/C=DE
!14:56:98:D3:FE:9C:CA:5A:31:6E:BC:81:D3:11:4E:00:90:A3:44:C2 S
```

Before entering a key into this file, you need to ensure its authenticity. How to do this depends on your organisation; your administrator might have already entered those keys which are deemed trustworthy enough into this file. Places where to look for the fingerprint of a root certificate are letters received from the CA or the website of the CA (after making 100% sure that this is indeed the website of that CA). You may want to consider allowing interactive updates of this file by using the see: [option --allow-mark-trusted]. This is however not as secure as maintaining this file manually. It is even advisable to change the permissions to read-only so that this file can't be

changed inadvertently.

As a special feature a line **include-default** will include a global list of trusted certificates (e.g. '*/etc/gnupg/trustlist.txt*'). This global list is also used if the local list is not available.

It is possible to add further flags after the **S** for use by the caller:

relax Relax checking of some root certificate requirements. As of now this flag allows the use of root certificates with a missing basicConstraints attribute (despite that it is a MUST for CA certificates) and disables CRL checking for the root certificate.

cm If validation of a certificate finally issued by a CA with this flag set fails, try again using the chain validation model.

sshcontrol

This file is used when support for the secure shell agent protocol has been enabled (see: [option --enable-ssh-support]). Only keys present in this file are used in the SSH protocol. You should backup this file.

The **ssh-add** tool may be used to add new entries to this file; you may also add them manually. Comment lines, indicated by a leading hash mark, as well as empty lines are ignored. An entry starts with optional whitespace, followed by the keygrip of the key given as 40 hex digits, optionally followed by the caching TTL in seconds and another optional field for arbitrary flags. A non-zero TTL overrides the global default as set by **--default-cache-ttl-ssh**.

The only flag support is **confirm**. If this flag is found for a key, each use of the key will pop up a pinentry to confirm the use of that key. The flag is automatically set if a new key was loaded into **gpg-agent** using the option **-c** of the **ssh-add** command.

The keygrip may be prefixed with a **!** to disable an entry entry.

The following example lists exactly one key. Note that keys available through a OpenPGP smartcard in the active smartcard reader are implicitly added to this list; i.e. there is no need to list them.

```
# Key added on: 2011-07-20 20:38:46
# Fingerprint: 5e:8d:c4:ad:e7:af:6e:27:8a:d6:13:e4:79:ad:0b:81
34B62F25E277CF13D3C6BCEBFD3F85D08F0A864B 0 confirm
```

private-keys-v1.d/

This is the directory where gpg-agent stores the private keys. Each key is stored in a file with the name made up of the keygrip and the suffix '*key*'. You should backup all files in this directory and take great care to keep this backup closed away.

Note that on larger installations, it is useful to put predefined files into the directory '*/etc/skel/.gnupg/*' so that newly created users start up with a working configuration. For existing users the a small helper script is provided to create these files (see: [addgnupghome]).

SIGNALS

A running **gpg-agent** may be controlled by signals, i.e. using the **kill** command to send a signal to the process.

Here is a list of supported signals:

SIGHUP
This signal flushes all cached passphrases and if the program has been started with a configuration file, the configuration file is read again. Only certain options are honored: **quiet**, **verbose**, **debug**, **debug-all**, **debug-level**, **no-grab**, **pinentry-program**, **default-cache-ttl**, **max-cache-ttl**, **ignore-cache-for-signing**, **allow-mark-trusted**, **disable-scdaemon**, and **disable-check-own-socket**. **scdaemon-program** is also supported but due to the current implementation, which calls the scdaemon only once, it is not of much use unless you manually kill the scdaemon.

SIGTERM
Shuts down the process but waits until all current requests are fulfilled. If the process has received 3 of these signals and requests are still pending, a shutdown is forced.

SIGINT
Shuts down the process immediately.

SIGUSR1
Dump internal information to the log file.

SIGUSR2
This signal is used for internal purposes.

SEE ALSO

gpg2(1), **gpgsm(1)**, **gpg-connect-agent(1)**, **scdaemon(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.