Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'gpg-card.1' command

$ man gpg-card.1

GPG-CARD(1) GNU Privacy Guard 2.2 GPG-CARD(1)

NAME

gpg-card - Administrate Smart Cards

SYNOPSIS

gpg-card [options]

gpg-card [options] command { -- command }

DESCRIPTION

The gpg-card is used to administrate smart cards and USB tokens. It provides a superset of features from gpg --card-edit an can be consid?ered a frontend to scdaemon which is a daemon started by gpg-agent to handle smart cards.

If gpg-card is invoked without commands an interactive mode is used.

If gpg-card is invoked with one or more commands the same commands as available in the interactive mode are run from the command line. These commands need to be delimited with a double-dash. If a double-dash or a shell specific character is required as part of a command the entire command needs to be put in quotes. If one of those commands returns an error the remaining commands are not anymore run unless the command was prefixed with a single dash.

A list of commands is available by using the command help and a brief description of each command is printed by using help CMD. See the sec?tion COMMANDS for a full description.

See the NOTES sections for instructions pertaining to specific cards or card applications.

COMMANDS

gpg-card understands the following commands, which have options of their own. The pseudo-option ?--? can be used to separate command options from arguments; if this pseudo option is used on the command line the entire command with options and arguments must be quoted, so that it is not mixed up with the ?--? as used on the command line to separate commands. Note that a short online help is available for all commands by prefixing them with ``help''. Command completion in the interactive mode is also supported.

AUTHENTICATE [--setkey] [--raw] [< file]|key]

AUTH Authenticate to the card. Perform a mutual autentication either by reading the key from file or by taking it from the command line as key. Without the option --raw the key is expected to be hex encoded. To install a new administration key --setkey is used; this requires a prior authentication with the old key. This is used with PIV cards.

CAFPR [--clear] N

Change the CA fingerprint number N of an OpenPGP card. N must be in the range 1 to 3. The option --clear clears the specified CA fingerprint N or all of them if N is 0 or not given.

FACTORY-RESET

Do a complete reset of some OpenPGP and PIV cards. This command deletes all data and keys and resets the PINs to their default. Don't worry, you need to confirm before the command proceeds.

FETCH Retrieve a key using the URL data object of an OpenPGP card or if that is missing using the stored fingerprint.

FORCESIG

Toggle the forcesig flag of an OpenPGP card.

GENERATE [--force] [--algo=algo{+algo2}] keyref

Create a new key on a card. Use --force to overwrite an existing key. Use "help" for algo to get a list of known algorithms. For OpenPGP cards several algos may be given. Note that the

OpenPGP key generation is done interactively unless --algo or keyref are given.

KDF-SETUP

Prepare the OpenPGP card KDF feature for this card.

LANG [--clear]

Change the language info for the card. This info can be used by applications for a personalized greeting. Up to 4 two-digit language identifiers can be entered as a preference. The option --clear removes all identifiers. GnuPG does not use this info.

LIST [--cards] [--apps] [--info] [--no-key-lookup] [n] [app]

L This command reads all information from the current card and display them in a human readable format. The first section shows generic information vaialable for all cards. The next section shows information pertaining to keys which depend on the actual card and application.

With n given select and list the n-th card; with app also given select that application. To select an app on the current card use "-" for n. The serial number of the card may be used in? stead of n.

The option --cards lists the serial numbers of available cards. The option --apps lists all card applications. The option --info selects a card and prints its serial number. The option --no-key-lookup suppresses the listing of matching OpenPGP or X.509 keys.

LOGIN [--clear] [< file]

Set the login data object of OpenPGP cards. If file is given the data is is read from that file. This allows to store binary data in the login field. The option --clear deletes the login data object.

NAME [--clear]

Set the name field of an OpenPGP card. With option --clear the stored name is cleared off the card.

PASSWD [--reset|--nullpin] [pinref]

Change or unblock the PINs. Note that in interactive mode and without a pinref a menu is presented for certain cards." In non-interactive mode and without a pinref a default value i used for these cards. The option --reset is used with TCOS cards to reset the PIN using the PUK or vice versa; the option --nullpin is used for these cards to set the intial PIN.

PRIVATEDO [--clear] n [< file]

Change the private data object n of an OpenPGP card. n must be in the range 1 to 4. If file is given the data is is read from that file. The option --clear clears the data.

QUIT

Q Stop processing and terminate gpg-card.

READCERT [--openpgp] certref > file

Read the certificate for key certref and store it in file. With option --openpgp an OpenPGP keyblock wrapped in a dedicated CMS content type (OID=1.3.6.1.4.1.11591.2.3.1) is expected and ex?tracted to file. Note that for current OpenPGP cards a certifi?cate may only be available at the certref "OPENPGP.3".

RESET Send a reset to the card daemon.

SALUTATION [--clear]

SALUT Change the salutation info for the card. This info can be used by applications for a personalized greeting. The option --clear removes this data object. GnuPG does not use this info.

UIF N [on|off|permanent]

Change the User Interaction Flag. That flags tells whether the confirmation button of a token shall be used. n must in the range 1 to 3. "permanent" is the same as "on" but the flag can't be changed anmore.

UNBLOCK

Unblock a PIN using a PUK or Reset Code. Note that OpenPGP cards prior to version 2 can't use this; instead the PASSWD can be used to set a new PIN.

URL [--clear]

Set the URL data object of an OpenPGP card. That data object can be used by by gpg's --fetch command to retrieve the full public key. The option --clear deletes the content of that data object.

VERIFY [chvid]

Verify the PIN identified by chvid or the default PIN.

WRITECERT certref < file

WRITECERT --openpgp certref [< file|fpr]

WRITECERT --clear certref

Write a certificate to the card under the id certref. The op?tion --clear removes the certificate from the card. The option --openpgp expects an OpenPGP keyblock and stores it encapsulated in a CMS container; the keyblock is taken from file or directly from the OpenPGP key identified by fingerprint fpr.

WRITEKEY [--force] keyref keygrip

Write a private key object identified by keygrip to the card un?der the id keyref. Option --force allows overwriting an exist?ing key.

YUBIKEY cmd args

Various commands pertaining to Yubikey tokens with cmd being:

LIST List supported and enabled Yubikey applications.

ENABLE usb|nfc|all [otp|u2f|opgp|piv|oath|fido2|all]

DISABLE

Enable or disable the specified or all applications on the given interface.

NOTES (OPENPGP)

The support for OpenPGP cards in gpg-card is not yet complete. For missing features, please continue to use gpg --card-edit.

NOTES (PIV)

GnuPG has support for PIV cards (``Personal Identity Verification'' as specified by NIST Special Publication 800-73-4). This section de?scribes how to initialize (personalize) a fresh Yubikey token featuring the PIV application (requires Yubikey-5). We assume that the creden?

tials have not yet been changed and thus are:

Authentication key

This is a 24 byte key described by the hex string 010203040506070801020304050607080102030405060708.

PIV Application PIN

This is the string 123456.

PIN Unblocking Key

This is the string 12345678.

See the example section on how to change these defaults. For production use it is important to use secure values for them. Note that the Authentication Key is not queried via the usual Pinentry dialog but needs to be entered manually or read from a file. The use of a dedicated machine to personalize tokens is strongly suggested.

To see what is on the card, the command list can be given. We will use the interactive mode in the following (the string gpg/card> is the prompt). An example output for a fresh card is:

```
  gpg/card> list
  Reader ...........: 1050:0407:X:0
  Card type ........: yubikey
  Card firmware ....: 5.1.2
  Serial number ....: D2760001240102010006090746250000
  Application type .: OpenPGP
  Version ..........: 2.1
  [...]
```

It can be seen by the ``Application type'' line that GnuPG selected the OpenPGP application of the Yubikey. This is because GnuPG assigns the highest priority to the OpenPGP application. To use the PIV application of the Yubikey several methods can be used:

With a Yubikey 5 or later the OpenPGP application on the Yubikey can be disabled:

```
  gpg/card> yubikey disable all opgp
  gpg/card> yubikey list
  Application  USB    NFC
  -----------------------
  OTP          yes    yes
  U2F          yes    yes
  OPGP          no     no
  PIV          yes    no
  OATH          yes    yes
  FIDO2         yes    yes
  gpg/card> reset
```

The reset is required so that the GnuPG system rereads the card. Note that disabled applications keep all their data and can at any time be re-enabled (use ?help yubikey?).

Another option, which works for all Yubikey versions, is to disable the support for OpenPGP cards in scdaemon. This is done by adding the line

```
  disable-application openpgp
```

to ?~/.gnupg/scdaemon.conf? and by restarting scdaemon, either by killing the process or by using ?gpgconf --kill scdaemon?. Finally the default order in which card applications are tried by scdaemon can be changed. For example to prefer PIV over OpenPGP it is sufficient to add

```
  application-priority piv
```

to ?~/.gnupg/scdaemon.conf? and to restart scdaemon. This has an ef? fect only on tokens which support both, PIV and OpenPGP, but does not hamper the use of OpenPGP only tokens.

With one of these methods employed the list command of gpg-card shows this:

```
  gpg/card> list
  Reader ...........: 1050:0407:X:0
  Card type ........: yubikey
  Card firmware ....: 5.1.2
  Serial number ....: FF020001008A77C1
  Application type .: PIV
  Version ..........: 1.0
  Displayed s/n ....: yk-9074625
  PIN usage policy .: app-pin
  PIN retry counter : - 3 -
  PIV authentication: [none]
        keyref .....: PIV.9A
  Card authenticat. : [none]
        keyref .....: PIV.9E
  Digital signature : [none]
        keyref .....: PIV.9C
  Key management ...: [none]
        keyref .....: PIV.9D
```

In case several tokens are plugged into the computer, gpg-card will show only one. To show another token the number of the token (0, 1, 2, ...) can be given as an argument to the list command. The command ?list --cards? prints a list of all inserted tokens.

Note that the ``Displayed s/n'' is printed on the token and also shown in Pinentry prompts asking for the PIN. The four standard key slots are always shown, if other key slots are initialized they are shown as well. The PIV authentication key (internal reference PIV.9A) is used to authenticate the card and the card holder. The use of the associated private key is protected by the Application PIN which needs to be provided once and the key can the be used until the card is reset or removed from the reader or USB port. GnuPG uses this key with its Secure Shell support. The Card authentication key (PIV.9E) is also known as the CAK and used to support physical access applications. The private key is not protected by a PIN and can thus immediately be used. The Digital signature key (PIV.9C) is used to digitally sign documents. The use of the associated private key is protected by the Application PIN which needs to be provided for each signing operation. The Key management key (PIV.9D) is used for encryption. The use of the associated private key is protected by the Application PIN which needs to be provided only once so that decryption operations can then be done until the card is reset or removed from the reader or USB port.

We now generate three of the four keys. Note that GnuPG does currently not use the the Card authentication key; however, that key is mandatory by the PIV standard and thus we create it too. Key generation requires that we authenticate to the card. This can be done either on the com?mand line (which would reveal the key):

```
  gpg/card> auth 010203040506070801020304050607080102030405060708
```

or by reading the key from a file. That file needs to consist of one LF terminated line with the hex encoded key (as above):

```
  gpg/card> auth < myauth.key
```

As usual ?help auth? gives help for this command. An error message is printed if a non-matching key is used. The authentication is valid un?til a reset of the card or until the card is removed from the reader or the USB port. Note that that in non-interactive mode the ?<? needs to be quoted so that the shell does not interpret it as a its own redi?rection symbol.

Here are the actual commands to generate the keys:

```
  gpg/card> generate --algo=nistp384 PIV.9A
  PIV card no. yk-9074625 detected
  gpg/card> generate --algo=nistp256 PIV.9E
  PIV card no. yk-9074625 detected
  gpg/card> generate --algo=rsa2048 PIV.9C
  PIV card no. yk-9074625 detected
```

If a key has already been created for one of the slots an error will be printed; to create a new key anyway the option ?--force? can be used. Note that only the private and public keys have been created but no certificates are stored in the key slots. In fact, GnuPG uses its own non-standard method to store just the public key in place of the the certificate. Other application will not be able to make use these keys until gpgsm or another tool has been used to create and store the re?spective certificates. Let us see what the list command now shows:

```
  gpg/card> list
  Reader ...........: 1050:0407:X:0
  Card type ........: yubikey
  Card firmware ....: 5.1.2
  Serial number ....: FF020001008A77C1
  Application type .: PIV
  Version ..........: 1.0
  Displayed s/n ....: yk-9074625
  PIN usage policy .: app-pin
  PIN retry counter : - 3 -
  PIV authentication: 213D1825FDE0F8240CB4E4229F01AF90AC658C2E
        keyref .....: PIV.9A  (auth)
        algorithm ..: nistp384
  Card authenticat. : 7A53E6CFFE7220A0E646B4632EE29E5A7104499C
        keyref .....: PIV.9E  (auth)
        algorithm ..: nistp256
  Digital signature : 32A6C6FAFCB8421878608AAB452D5470DD3223ED
        keyref .....: PIV.9C  (sign,cert)
        algorithm ..: rsa2048
  Key management ...: [none]
        keyref .....: PIV.9D
```

The primary information for each key is the keygrip, a 40 byte hex-string identifying the key. This keygrip is a unique identifier for the specific parameters of a key. It is used by gpg-agent and other parts of GnuPG to associate a private key to its protocol specific cer?tificate format (X.509, OpenPGP, or SecureShell). Below the keygrip the key reference along with the key usage capabilities are show. Fi?nally the algorithm is printed in the format used by {gpg}. At that point no other information is shown because for these new keys gpg won't be able to find matching certificates.

Although we could have created the Key management key also with the generate command, we will create that key off-card so that a backup ex?ists. To accomplish this a key needs to be created with either gpg or gpgsm or imported in one of these tools. In our example we create a self-signed X.509 certificate (exit the gpg-card tool, first):

```
  $ gpgsm --gen-key -o encr.crt
     (1) RSA
```

```
   (2) Existing key
   (3) Existing key from card
Your selection? 1
What keysize do you want? (3072) 2048
Requested keysize is 2048 bits
Possible actions for a RSA key:
   (1) sign, encrypt
   (2) sign
   (3) encrypt
Your selection? 3
Enter the X.509 subject name: CN=Encryption key for yk-9074625,O=example,C=DE
Enter email addresses (end with an empty line):
> otto@example.net
>
Enter DNS names (optional; end with an empty line):
>
Enter URIs (optional; end with an empty line):
>
Create self-signed certificate? (y/N) y
These parameters are used:
    Key-Type: RSA
    Key-Length: 2048
    Key-Usage: encrypt
    Serial: random
    Name-DN: CN=Encryption key for yk-9074625,O=example,C=DE
    Name-Email: otto@example.net
Proceed with creation? (y/N)
Now creating self-signed certificate.  This may take a while ...
gpgsm: about to sign the certificate for key: &34798AAFE0A7565088101CC4AE31C5C8C74461CB
gpgsm: certificate created
Ready.
$ gpgsm --import encr.crt
gpgsm: certificate imported
```

```
  gpgsm: total number processed: 1
  gpgsm:               imported: 1
```

Note the last step which imported the created certificate. If you you instead created a certificate signing request (CSR) instead of a self-signed certificate and sent this off to a CA you would do the same im?port step with the certificate received from the CA. Take note of the keygrip (prefixed with an ampersand) as shown during the certificate creation or listed it again using ?gpgsm --with-keygrip -k otto@exam?ple.net?. Now to move the key and certificate to the card start gpg-card again and enter:

```
  gpg/card> writekey PIV.9D 34798AAFE0A7565088101CC4AE31C5C8C74461CB
  gpg/card> writecert PIV.9D < encr.crt
```

If you entered a passphrase to protect the private key, you will be asked for it via the Pinentry prompt. On success the key and the cer?tificate has been written to the card and a list command shows:

```
  [...]
  Key management ...: 34798AAFE0A7565088101CC4AE31C5C8C74461CB
        keyref .....: PIV.9D  (encr)
        algorithm ..: rsa2048
        used for ...: X.509
          user id ..: CN=Encryption key for yk-9074625,O=example,C=DE
          user id ..: <otto@example.net>
```

In case the same key (identified by the keygrip) has been used for sev?eral certificates you will see several ``used for'' parts. With this the encryption key is now fully functional and can be used to decrypt messages encrypted to this certificate. Take care: the original key is still stored on-disk and should be moved to a backup medium. This can simply be done by copying the file ?34798AAFE0A7565088101CC4AE31C5C8C74461CB.key? from the directory ?~/.gnupg/private-keys-v1.d/? to the backup medium and deleting the file at its original place.

The final example is to create a self-signed certificate for digital signatures. Leave gpg-card using quit or by pressing Control-D and use gpgsm:

```
 $ gpgsm --learn
 $ gpgsm --gen-key -o sign.crt
 Please select what kind of key you want:
    (1) RSA
    (2) Existing key
    (3) Existing key from card
 Your selection? 3
 Serial number of the card: FF020001008A77C1
 Available keys:
    (1) 213D1825FDE0F8240CB4E4229F01AF90AC658C2E PIV.9A nistp384
    (2) 7A53E6CFFE7220A0E646B4632EE29E5A7104499C PIV.9E nistp256
    (3) 32A6C6FAFCB8421878608AAB452D5470DD3223ED PIV.9C rsa2048
    (4) 34798AAFE0A7565088101CC4AE31C5C8C74461CB PIV.9D rsa2048
 Your selection? 3
 Possible actions for a RSA key:
    (1) sign, encrypt
    (2) sign
    (3) encrypt
 Your selection? 2
 Enter the X.509 subject name: CN=Signing key for yk-9074625,O=example,C=DE
 Enter email addresses (end with an empty line):
 > otto@example.net
 >
 Enter DNS names (optional; end with an empty line):
 >
 Enter URIs (optional; end with an empty line):
 >
 Create self-signed certificate? (y/N)
 These parameters are used:
     Key-Type: card:PIV.9C
     Key-Length: 1024
     Key-Usage: sign
     Serial: random
     Name-DN: CN=Signing key for yk-9074625,O=example,C=DE
     Name-Email: otto@example.net
 Proceed with creation? (y/N) y
 Now creating self-signed certificate.  This may take a while ...
 gpgsm: about to sign the certificate for key: &32A6C6FAFCB8421878608AAB452D5470DD3223ED
 gpgsm: certificate created
 Ready.
 $ gpgsm --import sign.crt
 gpgsm: certificate imported
 gpgsm: total number processed: 1
 gpgsm:               imported: 1
```

The use of ?gpgsm --learn? is currently necessary so that gpg-agent knows what keys are available on the card. The need for this command will eventually be removed. The remaining commands are similar to the creation of an on-disk key. However, here we select the ?Digital signature? key. During the creation process you will be asked for the Application PIN of the card. The final step is to write the certificate to the card using gpg-card:

```
  gpg/card> writecert PIV.9C < sign.crt
```

By running list again we will see the fully initialized card:

```
  Reader ...........: 1050:0407:X:0
  Card type ........: yubikey
  Card firmware ....: 5.1.2
  Serial number ....: FF020001008A77C1
  Application type .: PIV
  Version ..........: 1.0
  Displayed s/n ....: yk-9074625
  PIN usage policy .: app-pin
  PIN retry counter : - [verified] -
  PIV authentication: 213D1825FDE0F8240CB4E4229F01AF90AC658C2E
        keyref .....: PIV.9A  (auth)
        algorithm ..: nistp384
  Card authenticat. : 7A53E6CFFE7220A0E646B4632EE29E5A7104499C
        keyref .....: PIV.9E  (auth)
        algorithm ..: nistp256
  Digital signature : 32A6C6FAFCB8421878608AAB452D5470DD3223ED
        keyref .....: PIV.9C  (sign,cert)
        algorithm ..: rsa2048
        used for ...: X.509
          user id ..: CN=Signing key for yk-9074625,O=example,C=DE
          user id ..: <otto@example.net>
  Key management ...: 34798AAFE0A7565088101CC4AE31C5C8C74461CB
        keyref .....: PIV.9D  (encr)
        algorithm ..: rsa2048
        used for ...: X.509
          user id ..: CN=Encryption key for yk-9074625,O=example,C=DE
          user id ..: <otto@example.net>
```

It is now possible to sign and to encrypt with this card using gpgsm and to use the ?PIV authentication? key with ssh:

```
  $ ssh-add -l
  384 SHA256:0qnJ0Y0ehWxKcx2frLfEljf6GCdlO55OZed5HqGHsaU cardno:yk-9074625 (ECDSA)
```

As usual use ssh-add with the uppercase ?-L? to list the public ssh key. To use the certificates with Thunderbird or Mozilla, please con? sult the Scute manual for details.

If you want to use the same PIV keys also for OpenPGP (for example on a Yubikey to avoid switching between OpenPGP and PIV), this is also pos? sible:

```
  $ gpgsm --learn
  $ gpg --full-gen-key
  Please select what kind of key you want:
     (1) RSA and RSA (default)
     (2) DSA and Elgamal
     (3) DSA (sign only)
     (4) RSA (sign only)
    (14) Existing key from card
```

```
Your selection? 14
Serial number of the card: FF020001008A77C1
Available keys:
  (1) 213D1825FDE0F8240CB4E4229F01AF90AC658C2E PIV.9A nistp384 (auth)
  (2) 7A53E6CFFE7220A0E646B4632EE29E5A7104499C PIV.9E nistp256 (auth)
  (3) 32A6C6FAFCB8421878608AAB452D5470DD3223ED PIV.9C rsa2048 (cert,sign)
  (4) 34798AAFE0A7565088101CC4AE31C5C8C74461CB PIV.9D rsa2048 (encr)
Your selection? 3
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key.
Real name:
Email address: otto@example.net
Comment:
You selected this USER-ID:
    "otto@example.net"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
gpg: key C3AFA9ED971BB365 marked as ultimately trusted
gpg: revocation certificate stored as '[...]D971BB365.rev'
public and secret key created and signed.
Note that this key cannot be used for encryption.  You may want to use
the command "--edit-key" to generate a subkey for this purpose.
pub   rsa2048 2019-04-04 [SC]
      7F899AE2FB73159DD68A1B20C3AFA9ED971BB365
uid                      otto@example.net
```

Note that you will be asked two times to enter the PIN of your PIV card. If you run gpg in --expert mode you will also ge given the op?tion to change the usage flags of the key. The next typescript shows how to add the encryption subkey:

```
$ gpg --edit-key 7F899AE2FB73159DD68A1B20C3AFA9ED971BB365
Secret key is available.
sec  rsa2048/C3AFA9ED971BB365
     created: 2019-04-04  expires: never       usage: SC
     card-no: FF020001008A77C1
     trust: ultimate      validity: ultimate
[ultimate] (1). otto@example.net
gpg> addkey
Secret parts of primary key are stored on-card.
Please select what kind of key you want:
   (3) DSA (sign only)
   (4) RSA (sign only)
   (5) Elgamal (encrypt only)
   (6) RSA (encrypt only)
  (14) Existing key from card
Your selection? 14
Serial number of the card: FF020001008A77C1
Available keys:
   (1) 213D1825FDE0F8240CB4E4229F01AF90AC658C2E PIV.9A nistp384 (auth)
   (2) 7A53E6CFFE7220A0E646B4632EE29E5A7104499C PIV.9E nistp256 (auth)
   (3) 32A6C6FAFCB8421878608AAB452D5470DD3223ED PIV.9C rsa2048 (cert,sign)
   (4) 34798AAFE0A7565088101CC4AE31C5C8C74461CB PIV.9D rsa2048 (encr)
Your selection? 4
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
```

```
  Key does not expire at all
  Is this correct? (y/N) y
  Really create? (y/N) y
  sec  rsa2048/C3AFA9ED971BB365
       created: 2019-04-04  expires: never       usage: SC
       card-no: FF020001008A77C1
       trust: ultimate      validity: ultimate
  ssb  rsa2048/7067860A98FCE6E1
       created: 2019-04-04  expires: never       usage: E
       card-no: FF020001008A77C1
  [ultimate] (1). otto@example.net
  gpg> save
```

Now you can use your PIV card also with gpg.

OPTIONS

gpg-card understands these options:

--with-colons

This option has currently no effect.

--status-fd n

Write special status strings to the file descriptor n. This program returns only 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.

--verbose

Enable extra informational output.

--quiet

Disable almost all informational output.

--version

Print version of the program and exit.

--help Display a brief help page and exit.

--no-autostart

Do not start the gpg-agent if it has not yet been started and its service is required. This option is mostly useful on ma?chines where the connection to gpg-agent has been redirected to another machines.

--no-history

In interactive mode the command line history is usually saved and restored to and from a file below the GnuPG home directory. This option inhibits the use of that file.

--agent-program file

Specify the agent program to be started if none is running. The default value is determined by running gpgconf with the option --list-dirs.

--gpg-program file

Specify a non-default gpg binary to be used by certain commands.

--gpgsm-program file

Specify a non-default gpgsm binary to be used by certain com?mands.

--chuid uid

Change the current user to uid which may either be a number or a name. This can be used from the root account to run gpg-card for another user. If uid is not the current UID a standard PATH is set and the envvar GNUPGHOME is unset. To override the lat?ter the option --homedir can be used. This option has only an effect when used on the command line. This option has currently no effect at all on Windows.

SEE ALSO

scdaemon(1)

GnuPG 2.3.3 2021-10-06 GPG-CARD(1)