NAME

virt-admin − daemon administration interface

SYNOPSIS

virt−admin [*OPTION*]... [*COMMAND_STRING*]

virt−admin [*OPTION*]... *COMMAND* [*ARG*]...

DESCRIPTION

The **virt−admin** program is the main administration interface for modifying the libvirt daemon configuration at runtime, changing daemon behaviour as well as for monitoring and managing all clients connected to the daemon.

The basic structure of most virt−admin usage is:

```
   virt−admin [OPTION]... <command> [ARG]...
```

Where *command* is one of the commands listed below. Any *command* starting with **#** is treated as a comment and silently ignored, all other unrecognized *commands* are diagnosed.

The **virt−admin** program can be used either to run one *COMMAND* by giving the command and its arguments on the shell command line, or a *COMMAND_STRING* which is a single shell argument consisting of multiple *COMMAND* actions and their arguments joined with whitespace and separated by semicolons or newlines between commands, where unquoted backslash−newline pairs are elided. Within *COMMAND_STRING*, virt−admin understands the same single, double, and backslash escapes as the shell, although you must add another layer of shell escaping in creating the single shell argument, and any word starting with unquoted # begins a comment that ends at newline. If no command is given in the command line, **virt−admin** will then start a minimal interpreter waiting for your commands, and the **quit** command will then exit the program.

The **virt−admin** program understands the following *OPTIONS*.

−c, **−−connect** *URI*

Connect to the specified *URI*, as if by the **connect** command, instead of the default connection.

−d, **−−debug** *LEVEL*

Enable debug messages at integer *LEVEL* and above. *LEVEL* can range from 0 to 4 (default). See the documentation of **VIRT_ADMIN_DEBUG** environment variable below for the description of each *LEVEL*.

−h, **−−help**

Ignore all other arguments, and behave as if the **help** command were given instead.

−l, **−−log** *FILE*

Output logging details to *FILE*.

−q, **−−quiet**

Avoid extra informational messages.

−v, **−−version[=short]**

Ignore all other arguments, and prints the version of the libvirt library virt−admin is coming from

−V, **−−version=long**

Ignore all other arguments, and prints the version of the libvirt library virt−admin is coming from.

NOTES

Running **virt−admin** requires root privileges due to the communications channels used to talk to the daemon. Consider changing the *unix_sock_group* ownership setting to grant access to specific set of users or modifying *unix_sock_rw_perms* permissions. Daemon configuration file provides more information about setting permissions.

GENERIC COMMANDS

The following commands are generic.

help

Syntax:

```
help [command−or−group]
```

This lists each of the virt−admin commands. When used without options, all commands are listed, one per line, grouped into related categories, displaying the keyword for each group.

To display detailed information for a specific command, use its name as the option.

quit, exit

Syntax:

```
quit
exit
```

quit this interactive terminal

version

Syntax:

```
version
```

will print out the version info about which libvirt library was this client built from. As opposed to *virsh* client, the output already includes the version of the daemon.

Example:

```
$ virt−admin version
Compiled against library: libvirt 1.2.21
Using library: libvirt 1.2.21
Running against daemon: 1.2.20
```

cd

Syntax:

```
cd [directory]
```

Will change current directory to *directory*. The default directory for the **cd** command is the home directory or, if there is no *HOME* variable in the environment, the root directory.

This command is only available in interactive mode.

pwd

Syntax:

```
pwd
```

Will print the current directory.

connect

Syntax:

```
connect [URI]
```

(Re)−Connect to a daemon's administrating server. The *URI* parameter specifies how to connect to the administrating server. If*LIBVIRT_ADMIN_DEF AULT_URI* or *uri_default* (see below) were set, *connect* is automatically issued every time a command that requires an active connection is executed. Note that this only applies if there is no connection at all or there is an inactive one.

To find the currently used URI, check the *uri* command documented below.

uri

Syntax:

```
uri
```

Prints the administrating server canonical URI, can be useful in shell mode. If no *uri* was specified, neither *LIBVIRT_ADMIN_DEFAULT_URI* environment variable nor *uri_default* option (libvirt−admin.conf) were set, libvirtd:///system is used.

DAEMON COMMANDS

The following commands allow one to monitor the daemon's state as well as directly change its internal configuration.

server−list

Syntax:

```
server−list
```

Lists all manageable servers contained within the daemon the client is currently connected to.

daemon−log−filters

Syntax:

```
daemon−log−filters [−−filters string]
```

When run without arguments, this returns the currently defined set of logging filters. Providing an argument will cause the command to define a new set of logging filters.

- *−−filters*

Define a new set of logging filters where multiple filters are delimited by space. Each filter must conform to the form described in detail by */etc/libvirt/libvirtd.conf* (section 'Logging filters').

Example:

To define a filter which suppresses all e.g. 'virObjectUnref' DEBUG messages, use the following:

```
$ virt−admin daemon−log−filters "4:util.object"
```

(Note the '.' symbol which can be used to more fine−grained filters tailored to specific modules, in contrast, to affect the whole directory containing several modules this would become "4:util"):

daemon−log−outouts

Syntax:

```
daemon−log−outputs [−−outputs string]
```

When run without arguments, this returns the currently defined set of logging outputs. Providing an argument will cause the command to define a new set of logging outputs.

- −−*outputs*

Define a new set of logging outputs where multiple outputs are delimited by space. Each output must conform to the form described in detail by */etc/libvirt/libvirtd.conf* (section 'Logging outputs').

Example:

To replace the current setting for logging outputs with one that writes to a file while logging errors only, the following could be used:

```
$ virt−admin daemon−log−outputs "4:file:<absolute_path_to_the_file>"
```

To define multiple outputs at once they need to be delimited by spaces:

```
$ virt−admin daemon−log−outputs "4:stderr 2:syslog:<msg_ident>"
```

SERVER COMMANDS

The following commands manipulate daemon's server internal configuration. The *server* is specified by its name.

server−threadpool−info

Syntax:

```
server−threadpool−info server
```

Retrieve server's threadpool attributes. These attributes include:

- *minWorkers* as the bottom limit to the number of active workers,
- *maxWorkers* as the top limit to the number of active workers,
- *nWorkers* as the current number of workers in the threadpool,
- *freeWorkers* as the current number of workers available for a task,
- *prioWorkers* as the current number of priority workers in the threadpool, and
- *jobQueueDepth* as the current depth of threadpool's job queue.

Background

Each daemon server utilizes a threadpool to accomplish tasks requested by clients connected to it. Every time a client request arrives to the server, it checks whether there is a worker available to accomplish the given task or it should create a new worker for the job (rather than being destroyed, the worker becomes free once the task is finished). Creating new workers, however, is only possible when the current number of workers is still below the configured upper limit. In addition to these 'standard' workers, a threadpool also contains a special set of workers called *priority* workers. Their purpose is to perform tasks that, unlike tasks carried out by normal workers, are within libvirt's full control and libvirt guarantees that such a task cannot hang, thus will always finish. An example of such a task this would be destroying a domain:

```
$ virsh destroy <domain>.
```

server−threadpool−set

Syntax:

```
server−threadpool−set server [−−min−workers count] [−−max−workers count] [−−pri
```

Change threadpool attributes on a server. Only a fraction of all attributes as described in *server−threadpool−info* is supported for the setter.

- *−−min−workers*

  The bottom limit to number of active workers in a threadpool.

- *−−max−workers*

  The upper limit to number of active workers in a threadpool. If used in combination with option *−−min−workers*, the value for the upper limit has to be greater than the value for the bottom limit, otherwise the command results in an error.

- *−−priority−workers*

  The current number of active priority workers in a threadpool.

server−clients−info

Syntax:

```
server−clients−info server
```

Get information about the current setting of limits regarding connections of new clients. This information comprises of the limits to the maximum number of clients connected to *server*, maximum number of clients waiting for authentication, in order to be connected to the server, as well as the current runtime values, more specifically, the current number of clients connected to *server* and the current number of clients waiting for authentication.

Example:

```
# virt−admin server−clients−info libvirtd
nclients_max        : 120
nclients            : 3
nclients_unauth_max : 20
nclients_unauth     : 0
```

server−clients−set

Syntax:

```
server−clients−set server [−−max−clients count] [−−max−unauth−clients count]
```

Set new client−related limits on *server*.

- *−−max−clients*

  Change the upper limit of the maximum overall number of clients connected to *server* to value **count**. The value for this limit has to be always greater than the value of *−−max−unauth−clients*.

- *−−max−unauth−clients*

  Change the upper limit of the maximum number of clients waiting for authentication, in order to be connected to *server*, to value **count**. The value for this limit has to be always lower than the value of *−−max−clients*.

server−update−tls

Syntax:

```
server−update−tls server
```

Update tls context on *server*.

- *server*

  Available servers on a daemon. Currently only supports 'libvirtd' or 'virtproxyd'.

CLIENT COMMANDS

The following commands provide management and monitoring of clients connected to one of daemon's available servers. Clients are specified by their numeric ID which is obtained by listing all clients connected to a specified server (see command **client−list**).

client−list

Syntax:

```
client−list server
```

Print a table showing the list of clients connected to <server>, also providing information about transport type used on client's connection (supported transports include **unix**, **tcp**, and **tls**), as well as providing information about client's connection time (system local time is used).

client−info

Syntax:

```
client−info server client
```

Retrieve identity information about *client* from *server*. The attributes returned may vary depending on the connection transport used. Transport−dependent attributes include local client process's pid, uid, user name, and group name, as well as socket address of the remote peer, see **Examples** below.

On the other hand, transport−independent attributes include client's SELinux context (if enabled on the host) and SASL username (if SASL authentication is enabled within daemon).

Examples:

```
# virt−admin client−info libvirtd 1
id             : 1
connection_time: 2016−05−03 13:27:04+0200
transport      : unix
readonly       : yes
unix_user_id   : 0
unix_user_name : root
unix_group_id  : 0
unix_group_name: root
unix_process_id: 10201

# virt−admin client−info libvirtd 2
id             : 2
connection_time: 2016−05−03 13:30:33+0200
transport      : tcp
readonly       : no
sock_addr      : 127.0.0.1:57060
```

client−disconnect

Syntax:

```
client−disconnect server client
```

Close a connection originating from *client*. The *server* argument specifies the name of the server *client* is currently connected to.

ENVIRONMENT

The following environment variables can be set to alter the behaviour of **virt−admin**

- VIRT_ADMIN_DEBUG=<0 to 4>

  Turn on verbose debugging of virt−admin commands. Valid levels are

  - VIRT_ADMIN_DEBUG=0

    DEBUG − Messages at ALL levels get logged
  - VIRT_ADMIN_DEBUG=1

    INFO − Logs messages at levels INFO, NOTICE, WARNING and ERROR
  - VIRT_ADMIN_DEBUG=2

    NOTICE − Logs messages at levels NOTICE, WARNING and ERROR
  - VIRT_ADMIN_DEBUG=3

    WARNING − Logs messages at levels WARNING and ERROR
  - VIRT_ADMIN_DEBUG=4

    ERROR − Messages at only ERROR level gets logged.
- VIRT_ADMIN_LOG_FILE=``LOGFILE``

  The file to log virt−admin debug messages.
- LIBVIRT_ADMIN_DEFAULT_URI

  The daemon whose admin server to connect to by default. Set this to a URI, in the same format as accepted by the **connect** option. This overrides the default URI set in any client config file.
- VIRT_ADMIN_HISTSIZE

  The number of commands to remember in the command history. The default value is 500.
- LIBVIRT_DEBUG=LEVEL

  Turn on verbose debugging of all libvirt API calls. Valid levels are

  - LIBVIRT_DEBUG=1

    Messages at level DEBUG or above
  - LIBVIRT_DEBUG=2

    Messages at level INFO or above
  - LIBVIRT_DEBUG=3

    Messages at level WARNING or above

- LIBVIRT_DEBUG=4

  Messages at level ERROR or above

For further information about debugging options consult *https://libvirt.org/logging.html*

AUTHORS

Please refer to the AUTHORS file distributed with libvirt.

BUGS

Please report all bugs you discover. This should be done via either:

1. the mailing list

   https://libvirt.org/contact.html

2. the bug tracker

   https://libvirt.org/bugs.html

Alternatively, you may report bugs to your software distributor / vendor.

COPYRIGHT

LICENSE

SEE ALSO

virsh(1), virt−xml−validate(1), virt−host−validate(1), *https://libvirt.org/*