NAME

lvm.conf — Configuration file for LVM2

SYNOPSIS

/etc/lvm/lvm.conf

DESCRIPTION

lvm.conf is loaded during the initialisation phase of **lvm**(8). This file can in turn lead to other files being loaded - settings read in later override earlier settings. File timestamps are checked between commands and if any have changed, all the files are reloaded.

The settings defined in lvm.conf can be overridden by any of these extended configuration methods:

direct config override on command line

The **−−config ConfigurationString** command line option takes the ConfigurationString as direct string representation of the configuration to override the existing configuration. The ConfigurationString is of exactly the same format as used in any LVM configuration file.

profile config

A profile is a set of selected customizable configuration settings that are aimed to achieve a certain characteristics in various environments or uses. It's used to override existing configuration. Normally, the name of the profile should reflect that environment or use.

There are two groups of profiles recognised: **command profiles** and **metadata profiles**.

The **command profile** is used to override selected configuration settings at global LVM command level - it is applied at the very beginning of LVM command execution and it is used throughout the whole time of LVM command execution. The command profile is applied by using the **−−commandprofile ProfileName** command line option that is recognised by all LVM2 commands.

The **metadata profile** is used to override selected configuration settings at Volume Group/Logical Volume level - it is applied independently for each Volume Group/Logical Volume that is being processed. As such, each Volume Group/Logical Volume can store the profile name used in its metadata so next time the Volume Group/Logical Volume is processed, the profile is applied automatically. If Volume Group and any of its Logical Volumes have different profiles defined, the profile defined for the Logical Volume is preferred. The metadata profile can be attached/detached by using the **lvchange** and **vgchange** commands and their **−−metadataprofile ProfileName** and **−−detachprofile** options or the **−−metadataprofile** option during creation when using **vgcreate** or **lvcreate** command. The **vgs** and **lvs** reporting commands provide **-o vg_profile** and **-o lv_profile** output options to show the metadata profile currently attached to a Volume Group or a Logical Volume.

The set of options allowed for command profiles is mutually exclusive when compared to the set of options allowed for metadata profiles. The settings that belong to either of these two sets can't be mixed together and LVM tools will reject such profiles.

LVM itself provides a few predefined configuration profiles. Users are allowed to add more profiles with different values if needed. For this purpose, there's the **command_profile_template.profile** (for command profiles) and **metadata_profile_template.profile** (for metadata profiles) which contain all settings that are customizable by profiles of certain type. Users are encouraged to copy these template profiles and edit them as needed. Alternatively, the **lvm dumpconfig −−file <ProfileName.profile> −−type profilable-command <section>** or **lvm dumpconfig −−file <ProfileName.profile> −−type profilable-metadata <section>** can be used to generate a configuration with profilable settings in either of the type for given section and save it to new ProfileName.profile (if the section is not specified, all profilable settings are reported).

The profiles are stored in /etc/lvm/profile directory by default. This location can be changed by using the **config/profile_dir** setting. Each profile configuration is stored in **ProfileName.profile** file in the profile directory. When referencing the profile, the **.profile** suffix is left out.

tag config

See **tags** configuration setting description below.

When several configuration methods are used at the same time and when LVM looks for the value of a particular setting, it traverses this **config cascade** from left to right:

direct config override on command line -> **command profile config** -> **metadata profile config** -> **tag config** -> **lvm.conf**

No part of this cascade is compulsory. If there's no setting value found at the end of the cascade, a default value is used for that setting. Use **lvm dumpconfig** to check what settings are in use and what the default values are.

SYNTAX

This section describes the configuration file syntax.

Whitespace is not significant unless it is within quotes. This provides a wide choice of acceptable indentation styles. Comments begin with # and continue to the end of the line. They are treated as whitespace.

Here is an informal grammar:

file = **value***

A configuration file consists of a set of values.

value = **section** | **assignment**

A value can either be a new section, or an assignment.

section = **identifier** '**{**' **value*** '**}**'

A section groups associated values together. If the same section is encountered multiple times, the contents of all instances are concatenated together in the order of appearance.
It is denoted by a name and delimited by curly brackets.

```
e.g.    backup {
                ...
        }
```

assignment = **identifier** '**=**' (**array** | **type**)

An assignment associates a type with an identifier. If the identifier contains forward slashes, those are interpreted as path delimiters. The statement **section/key = value** is equivalent to **section { key = value }**. If multiple instances of the same key are encountered, only the last value is used (and a warning is issued).
e.g. **level = 7**

array = '**[**' (**type** '**,**')* **type** '**]**' | '**[**' '**]**'

Inhomogeneous arrays are supported.
Elements must be separated by commas.
An empty array is acceptable.

type = **integer** | **float** | **string**

integer = [0-9]*
float = [0-9]*'**.**'[0-9]*
string = '**"**'.*'**"**'

Strings with spaces must be enclosed in double quotes, single words that start with a letter can be left unquoted.

SECTIONS

The sections that may be present in the file are:

devices — Device settings

dir — Directory in which to create volume group device nodes. Defaults to "/dev". Commands also accept this as a prefix on volume group names.

scan — List of directories to scan recursively for LVM physical volumes. Devices in directories outside this hierarchy will be ignored. Defaults to "/dev".

preferred_names — List of patterns compared in turn against all the pathnames referencing the same device in in the scanned directories. The pathname that matches the earliest pattern in the list is the one used in any output. As an example, if device-mapper multipathing is used, the following will select multipath device names:
devices { preferred_names = ["^/dev/mapper/mpath"] }

filter — List of patterns to apply to devices found by a scan. Patterns are regular expressions delimited by any character and preceded by **a** (for accept) or **r** (for reject). The list is traversed in order, and the first regex that matches determines if the device will be accepted or rejected (ignored). Devices that don't match any patterns are accepted. If you want to reject patterns that don't match, end the list with "r/.*/". If there are several names for the same device (e.g. symbolic links in /dev), if the first matching pattern in the list for any of the names is an **a** pattern, the device is accepted; otherwise if the first matching pattern in the list for any of the names is an **r** pattern it is rejected; otherwise it is accepted. As an example, to ignore /dev/cdrom you could use:
devices { filter=["r|cdrom|"] }

global_filter — Since "filter" might get overridden from the command line, it is not suitable for system-wide device filtering (udev rules, lvmetad). To hide devices from LVM-specific udev processing and/or from lvmetad, you need to set global_filter. The syntax is the same as for normal "filter" above. Devices that fail the global_filter are not even opened by LVM.

cache_dir — Persistent filter cache file directory. Defaults to "/etc/lvm/cache".

write_cache_state — Set to 0 to disable the writing out of the persistent filter cache file when **lvm** exits. Defaults to 1.

types — List of pairs of additional acceptable block device types found in /proc/devices together with maximum (non-zero) number of partitions (normally 16). By default, LVM2 supports ide, sd, md, loop, dasd, dac960, nbd, ida, cciss, ubd, ataraid, drbd, power2, i2o_block and iseries/vd. Block devices with major numbers of different types are ignored by LVM2. Example: **types = ["fd", 16]**. To create physical volumes on device-mapper volumes created outside LVM2, perhaps encrypted ones from **cryptsetup**, you'll need **types = ["device-mapper", 16]**. But if you do this, be careful to avoid recursion within LVM2. The figure for number of partitions is not currently used in LVM2 - and might never be.

sysfs_scan — If set to 1 and your kernel supports sysfs and it is mounted, sysfs will be used as a quick way of filtering out block devices that are not present.

md_component_detection — If set to 1, LVM2 will ignore devices used as components of software RAID (md) devices by looking for md superblocks. This doesn't always work satisfactorily e.g. if a device has been reused without wiping the md superblocks first.

md_chunk_alignment — If set to 1, and a Physical Volume is placed directly upon an md device, LVM2 will align its data blocks with the md device's stripe-width.

data_alignment_detection — If set to 1, and your kernel provides topology information in sysfs for the Physical Volume, the start of data area will be aligned on a multiple of the minimum_io_size or optimal_io_size exposed in sysfs. minimum_io_size is the smallest request the device can perform without incurring a read-modify-write penalty (e.g. MD's chunk size). optimal_io_size is the device's preferred unit of receiving I/O (e.g. MD's stripe width). minimum_io_size is used if optimal_io_size is undefined (0). If both **md_chunk_alignment** and

data_alignment_detection are enabled the result of **data_alignment_detection** is used.

data_alignment — Default alignment (in KB) of start of data area when creating a new Physical Volume using the **lvm2** format. If a Physical Volume is placed directly upon an md device and **md_chunk_alignment** or **data_alignment_detection** is enabled this parameter is ignored. Set to 0 to use the default alignment of 64KB or the page size, if larger.

data_alignment_offset_detection — If set to 1, and your kernel provides topology information in sysfs for the Physical Volume, the start of the aligned data area of the Physical Volume will be shifted by the alignment_offset exposed in sysfs.

To see the location of the first Physical Extent of an existing Physical Volume use **pvs −o +pe_start** . It will be a multiple of the requested **data_alignment** plus the alignment_offset from **data_alignment_offset_detection** (if enabled) or the pvcreate commandline.

disable_after_error_count — During each LVM operation errors received from each device are counted. If the counter of a particular device exceeds the limit set here, no further I/O is sent to that device for the remainder of the respective operation. Setting the parameter to 0 disables the counters altogether.

pv_min_size — Minimal size (in KB) of the block device which can be used as a PV. In clustered environment all nodes have to use the same value. Any value smaller than 512KB is ignored. Up to and include version 2.02.84 the default was 512KB. From 2.02.85 onwards it was changed to 2MB to avoid floppy drives by default.

issue_discards — Issue discards to a logical volumes's underlying physical volume(s) when the logical volume is no longer using the physical volumes' space (e.g. lvremove, lvreduce, etc). Discards inform the storage that a region is no longer in use. Storage that supports discards advertise the protocol specific way discards should be issued by the kernel (TRIM, UNMAP, or WRITE SAME with UNMAP bit set). Not all storage will support or benefit from discards but SSDs and thinly provisioned LUNs generally do. If set to 1, discards will only be issued if both the storage and kernel provide support.

allocation — Space allocation policies

cling_tag_list — List of PV tags matched by the **cling** allocation policy.

When searching for free space to extend an LV, the **cling** allocation policy will choose space on the same PVs as the last segment of the existing LV. If there is insufficient space and a list of tags is defined here, it will check whether any of them are attached to the PVs concerned and then seek to match those PV tags between existing extents and new extents.

The @ prefix for tags is required. Use the special tag "@*" as a wildcard to match any PV tag and so use all PV tags for this purpose.

For example, LVs are mirrored between two sites within a single VG. PVs are tagged with either @site1 or @site2 to indicate where they are situated and these two PV tags are selected for use with this allocation policy:

cling_tag_list = ["@site1", "@site2"]

cache_pool_cachemode — Cache mode for new cache pools.

This is the default cache mode a new cache pool will be given. Valid cache modes are: **writethrough** - Data blocks are immediately written from the cache to disk. **writeback** - Data blocks are written from the cache back to disk after some delay to improve performance.

log — Default log settings

file — Location of log file. If this entry is not present, no log file is written.

overwrite — Set to 1 to overwrite the log file each time a tool is invoked. By default tools append messages to the log file.

level — Log level (0-9) of messages to write to the file. 9 is the most verbose; 0 should produce no output.

verbose — Default level (0-3) of messages sent to stdout or stderr. 3 is the most verbose; 0 should produce the least output.

silent — Set to 1 to suppress all non-essential tool output. When set, display and reporting tools will still write the requested device properties to standard output, but messages confirming that something was or wasn't changed will be reduced to the 'verbose' level and not appear unless −v is supplied.

syslog — Set to 1 (the default) to send log messages through syslog. Turn off by setting to 0. If you set to an integer greater than one, this is used - unvalidated - as the facility. The default is LOG_USER. See /usr/include/sys/syslog.h for safe facility values to use. For example, LOG_LOCAL0 might be 128.

indent — When set to 1 (the default) messages are indented according to their severity, two spaces per level. Set to 0 to turn off indentation.

command_names — When set to 1, the command name is used as a prefix for each message. Default is 0 (off).

prefix — Prefix used for all messages (after the command name). Default is two spaces.

activation — Set to 1 to log messages while devices are suspended during activation. Only set this temporarily while debugging a problem because in low memory situations this setting can cause your machine to lock up.

backup — Configuration for metadata backups.

archive_dir — Directory used for automatic metadata archives. Backup copies of former metadata for each volume group are archived here. Defaults to "/etc/lvm/archive".

backup_dir — Directory used for automatic metadata backups. A single backup copy of the current metadata for each volume group is stored here. Defaults to "/etc/lvm/backup".

archive — Whether or not tools automatically archive existing metadata into **archive_dir** before making changes to it. Default is 1 (automatic archives enabled). Set to 0 to disable. Disabling this might make metadata recovery difficult or impossible if something goes wrong.

backup — Whether or not tools make an automatic backup into **backup_dir** after changing metadata. Default is 1 (automatic backups enabled). Set to 0 to disable. Disabling this might make metadata recovery difficult or impossible if something goes wrong.

retain_min — Minimum number of archives to keep. Defaults to 10.

retain_days — Minimum number of days to keep archive files. Defaults to 30.

shell — LVM2 built-in readline shell settings

history_size — Maximum number of lines of shell history to retain (default 100) in $HOME/.lvm_history

global — Global settings

test — If set to 1, run tools in test mode i.e. no changes to the on-disk metadata will get made. It's equivalent to having the -t option on every command.

activation — Set to 0 to turn off all communication with the device-mapper driver. Useful if you want to manipulate logical volumes while device-mapper is not present in your kernel.

proc — Mount point of proc filesystem. Defaults to /proc.

umask — File creation mask for any files and directories created. Interpreted as octal if the first digit is zero. Defaults to 077. Use 022 to allow other users to read the files by default.

format — The default value of **−−metadatatype** used to determine which format of metadata to use when creating new physical volumes and volume groups. **lvm1** or **lvm2**.

fallback_to_lvm1 — Set this to 1 if you need to be able to switch between 2.4 kernels using LVM1 and kernels including device-mapper. The LVM2 tools should be installed as normal and the LVM1 tools should be installed with a .lvm1 suffix e.g. vgscan.lvm1. If an LVM2 tool is then run but unable to communicate with device-mapper, it will automatically invoke the equivalent LVM1 version of the tool. Note that for LVM1 tools to manipulate physical volumes and volume groups created by LVM2 you must use **−−metadataformat lvm1** when creating them.

library_dir — A directory searched for LVM2's shared libraries ahead of the places **dlopen** (3) searches.

format_libraries — A list of shared libraries to load that contain code to process different formats of metadata. For example, liblvm2formatpool.so is needed to read GFS pool metadata if LVM2 was configured **−−with-pool=shared**.

locking_type — What type of locking to use. 1 is the default, which use flocks on files in **locking_dir** (see below) to avoid conflicting LVM2 commands running concurrently on a single machine. 0 disables locking and risks corrupting your metadata. If set to 2, the tools will load the external **locking_library** (see below). If the tools were configured **−−with-cluster=internal** (the default) then 3 means to use built-in cluster-wide locking. Type 4 enforces read-only metadata and forbids any operations that might want to modify Volume Group metadata. All changes to logical volumes and their states are communicated using locks.

wait_for_locks — When set to 1, the default, the tools wait if a lock request cannot be satisfied immediately. When set to 0, the operation is aborted instead.

locking_dir — The directory LVM2 places its file locks if **locking_type** is set to 1. The default is **/var/lock/lvm**.

locking_library — The name of the external locking library to load if **locking_type** is set to 2. The default is **liblvm2clusterlock.so**. If you need to write such a library, look at the lib/locking source code directory.

use_lvmetad — Whether to use (trust) a running instance of lvmetad. If this is set to 0, all commands fall back to the usual scanning mechanisms. When set to 1 **and** when lvmetad is running (it is not auto-started), the volume group metadata and PV state flags are obtained from the lvmetad instance and no scanning is done by the individual commands. In a setup with lvmetad, lvmetad udev rules **must** be set up for LVM to work correctly. Without proper udev rules, all changes in block device configuration will be **ignored** until a manual 'pvscan −−cache' is performed.
If lvmetad has been running while use_lvmetad was 0, it **MUST** be stopped before changing use_lvmetad to 1 and started again afterwards.

tags — Host tag settings

hosttags — If set to 1, create a host tag with the machine name. Setting this to 0 does nothing, neither creating nor destroying any tag. The machine name used is the nodename as returned by **uname** (2).

Additional host tags to be set can be listed here as subsections. The @ prefix for tags is optional. Each of these host tag subsections can contain a **host_list** array of host names. If any one of these entries matches the machine name exactly then the host tag gets defined on this particular host, otherwise it doesn't.

After lvm.conf has been processed, LVM2 works through each host tag that has been defined in turn, and if there is a configuration file called lvm_<**host_tag**>.conf it attempts to load it. The activation/volume_list, devices/filter and devices/types settings are merged (these all are lists), otherwise any settings read in override settings found in earlier files. Any additional host tags defined get appended to the search list, so in turn they can lead to further configuration files being processed. Use **lvm dumpconfig** to check the result of config file processing.

The following example always sets host tags **tag1** and sets **tag2** on machines fs1 and fs2:

tags { tag1 { } tag2 { host_list = ["fs1", "fs2"] } }

These options are useful if you are replicating configuration files around a cluster. Use of **hosttags = 1** means every machine can have static and identical local configuration files yet use different settings and activate different logical volumes by default. See also **volume_list** below and **−−addtag** in **lvm** (8).

activation — Settings affecting device-mapper activation

missing_stripe_filler — When activating an incomplete logical volume in partial mode, this option dictates how the missing data is replaced. A value of "error" will cause activation to create error mappings for the missing data, meaning that read access to missing portions of the volume will result in I/O errors. You can instead also use a device path, and in that case this device will be used in place of missing stripes. However, note that using anything other than "error" with mirrored or snapshotted volumes is likely to result in data corruption. For instructions on how to create a device that always returns zeros, see **lvcreate** (8).

mirror_region_size — Unit size in KB for copy operations when mirroring.

readahead — Used when there is no readahead value stored in the volume group metadata. Set to **none** to disable readahead in these circumstances or **auto** to use the default value chosen by the kernel.

reserved_memory, **reserved_stack** — How many KB to reserve for LVM2 to use while logical volumes are suspended. If insufficient memory is reserved before suspension, there is a risk of machine deadlock.

process_priority — The nice value to use while devices are suspended. This is set to a high priority so that logical volumes are suspended (with I/O generated by other processes to those logical volumes getting queued) for the shortest possible time.

volume_list — This acts as a filter through which all requests to activate a logical volume on this machine are passed. A logical volume is only activated if it matches an item in the list. Tags must be preceded by @ and are checked against all tags defined in the logical volume and volume group metadata for a match. @* is short-hand to check every tag set on the host machine (see **tags** above). Logical volume and volume groups can also be included in the list by name e.g. vg00, vg00/lvol1. If this setting is not present but at least one host tag is defined then a default single-entry list containing @* is assumed.

auto_activation_volume_list — This acts as a filter through which all requests to autoactivate a logical volume on this machine are passed. A logical volume is autoactivated if it matches an item in the list. Volumes must also pass the **volume_list** filter, if present. Tags must be preceded by @ and are checked against all tags defined in the logical volume and volume group metadata for a match. @* is short-hand to check every tag set on the host machine (see **tags** above). Logical volume and volume groups can also be included in the list by name e.g. vg00, vg00/lvol1.

read_only_volume_list — This acts as a filter through which all requests to activate a logical volume on this machine are passed. A logical volume is activated in read-only mode (instead of read-write) if it matches an item in the list. Volumes must first pass the **volume_list** filter, if present. Tags must be preceded by @ and are checked against all tags defined in the logical volume and volume group metadata for a match. @* is short-hand to check every tag set on the host machine (see **tags** above). Logical volume and volume groups can also be included in the list by name e.g. vg00, vg00/lvol1.

metadata — Advanced metadata settings

pvmetadatacopies — When creating a physical volume using the LVM2 metadata format, this is the default number of copies of metadata to store on each physical volume. Currently it can be set to 0, 1 or 2. The default is 1. If set to 2, one copy is placed at the beginning of the disk and the other is placed at the end. It can be overridden on the command line with **−−pvmetadatacopies** (see **pvcreate**). If creating a volume group with just one physical volume, it's a good idea to have 2 copies. If creating a large volume group with many physical volumes, you may decide that 3 copies of the metadata is sufficient, i.e. setting it to 1 on three of the physical volumes, and 0 on the rest. Every volume group must contain at least one physical volume with at least 1 copy of the metadata (unless using the text files described below). The disadvantage of having lots of copies is that every time the tools access the volume group, every copy of the metadata has to be accessed, and this slows down the tools.

pvmetadatasize — Approximate number of sectors to set aside for each copy of the metadata. Volume groups with large numbers of physical or logical volumes, or volumes groups containing complex logical volume structures will need additional space for their metadata. The metadata areas are treated as circular buffers, so unused space becomes filled with an archive of the most recent previous versions of the metadata.

pvmetadataignore When creating a physical volume using the LVM2 metadata format, this states whether metadata areas should be ignored. The default is "n". If metadata areas on a physical volume are ignored, LVM will not not store metadata in the metadata areas present on newly created Physical Volumes. The option can be overridden on the command line with **−−metadataignore** (See **pvcreate** and **pvchange**). Metadata areas cannot be created or extended after Logical Volumes have been allocated on the device. If you do not want to store metadata on this device, it is still wise always to allocate a metadata area (use a non-zero value for **−−pvmetadatacopies**) in case you need it in the future and to use this option to instruct LVM2 to ignore it.

vgmetadatacopies — When creating a volume group using the LVM2 metadata format, this is the default number of copies of metadata desired across all the physical volumes in the volume group. If set to a non-zero value, LVM will automatically set or clear the metadataignore flag on the physical volumes (see **pvcreate** and **pvchange −−metadataignore**) in order to achieve the desired number of metadata copies. An LVM command that adds or removes physical volumes (for example, **vgextend**, **vgreduce**, **vgsplit**, or **vgmerge**), may cause LVM to automatically set or clear the metadataignore flags. Also, if physical volumes go missing or reappear, or a new number of copies is explicitly set (see **vgchange −−vgmetadatacopies**), LVM may adjust the metadataignore flags. Set **vgmetadatacopies** to 0 instructs LVM not to set or clear the metadataignore flags automatically. You may set a value larger than the sum of all metadata areas on all physical volumes. The value can be overridden on the command line with **−−vgmetadatacopies** for various commands (for example, **vgcreate** and **vgchange**), and can be queryied with the **vg_mda_copies** field of **vgs**. This option is useful for volume groups containing large numbers of physical volumes with metadata as it may be used to minimize metadata read and write overhead.

dirs — List of directories holding live copies of LVM2 metadata as text files. These directories must not be on logical volumes. It is possible to use LVM2 with a couple of directories here, preferably on different (non-logical-volume) filesystems and with no other on-disk metadata, **pvmetadatacopies = 0**. Alternatively these directories can be in addition to the on-disk metadata areas. This feature was created during the development of the LVM2 metadata before the new on-disk metadata areas were designed and no longer gets tested. It is not supported under low-memory conditions, and it is important never to edit these metadata files unless you fully understand how things work: to make changes you should always use the tools as normal, or else vgcfg-backup, edit backup, vgcfgrestore.

FILES

/etc/lvm/lvm.conf
/etc/lvm/archive
/etc/lvm/backup
/etc/lvm/cache/.cache
/var/lock/lvm

SEE ALSO

lvm(8), **umask**(2), **uname**(2), **dlopen**(3), **syslog**(3), **syslog.conf**(5)