NAME

lvmreport — LVM reporting and related features

DESCRIPTION

LVM uses single reporting infrastructure that sets standard on LVM command's output and it provides wide range of configuration settings and command line options to customize report and filter the report's output.

Categorization based on reporting facility

Based on functionality, commands which make use of the reporting infrastructure are divided in two groups:

Report-oriented

These commands inform about current LVM state and their primary role is to display this information in compendious way. To make a distinction, we will name this report as **main report**. The set of report-only commands include: pvs, vgs, lvs, pvdisplay, vgdisplay, lvdisplay, lvm devtypes, lvm fullreport. For further information about main report, see **main report specifics**.

Processing-oriented

These commands are responsible for changing LVM state and they do not contain any main report as identified for report-oriented commands, they only perform some kind of processing. The set of processing-oriented commands includes: pvcreate, vgcreate, lvcreate, pvchange, vgchange, lvchange, pvremove, vgremove, lvremove, pvresize, vgextend, vgreduce, lvextend, lvreduce, lvresize, lvrename, pvscan, vgscan, lvscan, pvmove, vgcfgbackup, vgck, vgconvert, vgexport, vgimport, vgmknodes.

If enabled, so called **log report** is either displayed solely (for processing-oriented commands) or in addition to main report (for report-oriented commands). The log report contains a log of operations, messages and per-object status with complete object identification collected during LVM command execution. See **log report specifics** for more information about this report type.

Terms

When describing reporting functionality and features in this text, we will use terms **row** and **column**. By row we mean series of values reported for single entity (for example single PV, VG or LV). Each value from the row then belongs to a column of certain type. The columns have **column headings** which are short descriptions for the columns. The columns are referenced by **column names**. Please note that this text is also using term **field** interchangeably with the term **column**. Most of the time the term columns is abbreviated as **col** in configuration.

Common report configuration settings and command line options

There are common configuration settings and command line options which apply to both **main report** and **log report**. Following lists contain all of them, separated into groups based on their use.

Common configuration settings:

- Changing report output format, composition and other output modifiers:
  - global/units
  - global/suffix
  - report/output_format
  - report/compact_output

  - report/compact_output_cols
  - report/aligned
  - report/headings
  - report/separator
  - report/list_item_separator
  - report/prefixes
  - report/quoted
  - report/columns_as_rows
  - report/binary_values_as_numeric
  - report/time_format
  - report/mark_hidden_devices
  - report/two_word_unknown_device

- Special settings
  - report/buffered

This document does not describe these settings in more detail − if you need detailed information, including values which are accepted for the settings, please run **lvmconfig −−type default −−withcomments <setting>**. There are more configuration settings in addition to the common set listed above, but they are specific to either **main report** or **log report**, see **main report specifics** and **log report specifics** for these settings. Besides configuring reports globally by using configuration settings, there are also command line options you can use to extend, override or further specify the report configuration.

Common command line options:

- Definition of the set set of fields to use
  - --options|–o FieldSet
    Field set to use. See **main report specifics** and **log report specifics** for information about field sets configured with global configuratin settings that this option overrides.
  - --options|–o+ FieldSet
    Fields to include to current field set. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option extends.
  - --options|–o- FieldSet
    Fields to exclude from current field set. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option reduces.
  - --options|–o# FieldSet
    Compaction of unused fields. Overrides report/compact_output_cols configuration setting.

- Sorting

- --sort|–O+ FieldSet
  Fields to sort by in ascending order. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option overrides.
- --sort|–O- FieldSet
  Fields to sort by in descending order. See **main report specifics** and **log report specifics** for information about fields sets configured with global configuration settings that this options overrides.

• Selection

- --select|–S Selection
  Define selection criteria for report output. For **log report**, this also overrides log/command_log_selection configuration setting, see also **log report specifics**.

• Changing output format and composition

- --reportformat
  Overrides report/output_format configuration setting.
- --aligned
  Overrides report/aligned configuration setting.
- --binary
  Overrides report/binary_values_as_numeric configuration setting.
- --nameprefixes
  Overrides report/prefixes configuration setting.
- --noheadings
  Overrides report/noheadings configuration setting.
- --nosuffix
  Overrides global/suffix configuration setting.
- --rows
  Overrides report/columns_as_rows configuration setting.
- --separator
  Overrides report/separator configuration setting.
- --units
  Overrides global/units configuration setting.
- --unquoted
  Overrides report/quoted configuration setting.

• Special options

- --configreport **ReportName**
  This defines the **ReportName** for which any subsequent –o––columns, -O––sort or –S––select applies to. See also **main report specifics** and **log report specifics** for possible **ReportName** values.
- --logonly
  When an LVM command contains both **main report** and **log report**, this option suppresses the **main report** output and it causes the **log report** output to be displayed only.
- --unbuffered
  Overrides report/bufffered configuration setting.

The **FieldSet** mentioned in the lists above is a set of field names where each field name is delimited by "," character. Field set definition, sorting and selection may be repeated on command line (−o+/−o- includes/excludes fields to/from current list, for all the other repeatable options, the last value typed for the option on the command line is used). The **Selection** is a string with **selection criteria**, see also **Selection** paragraph below for more information about constructing these criteria.

Main report specifics

The **main report** currently encompasses these distinct subtypes, referenced by their name - **ReportName** as listed below. The command in parenthesis is representative command that uses the main report subtype by default. Each subtype has its own configuration setting for global field set definition as well as sort field definition (listed below each individual **ReportName**):

- **pv** representing report about Physical Volumes (**pvs**)
  - report/pvs_cols
  - report/pvs_sort
- **pvseg** representing report about Physical Volume Segments (**pvs −−segments**)
  - report/pvseg_cols
  - report/pvseg_sort
- **vg** representing report about Volume Groups (**vgs**)
  - report/vgs_cols
  - report/vgs_sort
- **lv** representing report about Logical Volumes (**lvs**)
  - report/lvs_cols
  - report/lvs_sort
- **seg** representing report about Logical Volume Segments (**lvs −−segments**)
  - report/segs_cols
  - report/segs_sort
- **full** representing report combining all of the above as a whole (**lvm fullreport**)
  - report/pvs_cols_full
  - report/pvs_sort_full
  - report/pvsegs_cols_full
  - report/pvseg_sort_full
  - report/vgs_cols_full
  - report/vgs_sort_full
  - report/lvs_cols_full
  - report/lvs_sort_full

- report/segs_cols_full
- report/segs_sort_full

- **devtype** representing report about device types (**lvm devtypes**)
  - report/devtypes_cols
  - report/devtypes_sort

Use **pvs, vgs, lvs −o help** or **lvm devtypes −o help** to get complete list of fields that you can use for main report. The list of fields in the help output is separated in groups based on which report type they belong to. Note that LVM can change final report type used if fields from different groups are combined together. Some of these combinations are not allowed in which case LVM will issue an error.

For all main report subtypes except **full**, it's not necessary to use **−−configreport ReportName** to denote which report any subsequent **−o, −O or −S** option applies to as they always apply to the single main report type. Currently, **lvm fullreport** is the only command that includes more than one **main report** subtype. Therefore, the −−configreport is particularly suitable for the full report if you need to configure each of its subreports in a different way.

Log report specifics

You can enable log report with **log/report_command_log** configuration setting − this functionality is disabled by default. The **log report** contains a log collected during LVM command execution and then the log is displayed just like any other report known from main report. There is only one log report subtype as shown below together with related configuration settings for fields, sorting and selection:

- **log** representing log report
  - log/command_log_cols
  - log/command_log_sort
  - log/command_log_selection

You always need to use **−−configreport log** together with **−o−−options, -O−−sort or −S−−selection** to override configuration settings directly on command line for **log report**. When compared to **main report**, in addition to usual configuration settings for report fields and sorting, the **log report** has also configuration option for selection - **report/command_log_selection**. This configuration setting is provided for convenience so it's not necessary to use **−S−−select** on command line each time an LVM command is executed and we need the same selection criteria to be applied for **log report**. Default selection criteria used for **log report** are **log/command_log_selection="!(log_type=status && message=success)"**. This means that, by default, **log report** doesn't display status messages about successful operation and it displays only rows with error, warning, print-type messages and messages about failure states (for more information, see **log report content** below).

Log report coverage
Currently, when running LVM commands directly (not in LVM shell), the log report covers command's **processing stage** which is the moment when LVM entities are iterated and processed one by one. It does not cover any command initialization nor command finalization stage. If there is any message issued out of log report's coverage range, such message goes directly to output, bypassing the **log report**. By default, that is **standard error output** for error and warning messages and **standard output** for common print-like messages.

When running LVM commands in **LVM shell**, the log report covers the whole LVM command's execution, including command's **processing** as well as **initialization** and **finalization stage**. So from this point of view, the log report coverage is complete for executed LVM commands. Note that there are still a few moments when LVM shell needs to initialize itself before it even enters the main loop in which it executes LVM commands. Also, there is a moment when **LVM shell** needs to prepare **log report** properly for next command executed in the shell and then, after the command's run, the shell needs to display the log report for that recently executed command. If there is a failure or any other message issued during this time, the LVM will bypass **log report** and display messages on output directly.

For these reasons and for completeness, it's not possible to rely fully on **log report** as the only indicator of LVM command's status and the only place where all messages issued during LVM command execution are collected. You always need to check whether the command has not failed out of log report's range by checking the non-report output too.

To help with this, LVM can separate output which you can then redirect to any **custom file descriptor** that you prepare before running an LVM command or LVM shell and then you make LVM to use these file descriptors for different kinds of output by defining environment variables with file descriptor numbers. See also **LVM_OUT_FD**, **LVM_ERR_FD** and **LVM_REPORT_FD** environment variable description in **lvm**(8) man page.

Also note that, by default, reports use the same file descriptor as common print-like messages, which is **standard output**. If you plan to use **log report** in your scripts or any external tool, you should use **LVM_OUT_FD**, **LVM_ERR_FD** and **LVM_REPORT_FD** to separate all output types to different file descriptors. For example, with bash, that would be:

```
LVM_OUT_FD=3 LVM_ERR_FD=4 LVM_REPORT_FD=5 <lvm command> 3>out_file
4>err_file 5>report_file
```

Where the <lvm_command> is either direct LVM command or LVM shell. You can collect all three types of output in particular files then.

Log report content

Each item in the log report consists of these set of fields providing various information:

- Basic information (mandatory):
  - log_seq_num
    Item sequence number. The sequence number is unique for each log item and it increases in the order of the log items as they appeared during LVM command execution.
  - log_type
    Type of log for the item. Currently, these types are used:

    status for any status information that is logged

    print for any common message printed while the log is collected

    error for any error message printed while the log is collected

    warn for any warning message printed while the log is collected
  - log_context
    Context of the log for the item. Currently, two contexts are identified:

    shell for the log collected in the outermost code before and after executing concrete LVM commands

processing for the log collected while processing LVM entities during LVM command execution

- Message (mandatory):
  - log_message
    Any message associated with current item. For **status** log type, the message contains either **success** or **failure** denoting current state. For **print**, **error** and **warn** log types, the message contains the exact message of that type that got issued.

- Object information (used only if applicable):
  - log_object_type field
    Type of the object processed. Currently, these object types are recognized:

    cmd for command as a whole

    orphan for processing group of PVs not in any VG yet

    pv for PV processing

    label for direct PV label processing (without VG metadata)

    vg for VG processing

    lv for LV processing

  - log_object_name
    Name of the object processed.

  - log_object_id
    ID of the object processed.

  - log_object_group
    A group where the processed object belongs to.

  - log_object_group_id
    An ID of a group where the processed object belongs to.

- Numeric status (used only if applicable)
  - log_errno
    Error number associated with current item.
  - log_ret_code
    Rreturn code associated with current item.

You can also run **<lvm_command> −−configreport log −o help** to to display complete list of fields that you may use for the **log report**.

Selection

Selection is used for a report to display only rows that match **selection criteria**. All rows are displayed with the additional **selected** field (**−o selected**) displaying 1 if the row matches the *Selection* and 0 otherwise. The **selection criteria** are a set of **statements** combined by **logical and grouping operators**. The

statement consists of a **field** name for which a set of valid **values** is defined using **comparison operators**. For complete list of fields names that you can use in selection, see the output of **<lvm_command> −S help**. The help output also contains type of values that each field displays enclosed in brackets.

List of operators recognized in selection criteria

- Comparison operators (cmp_op)

  =~ matching regular expression.

  !~ not matching regular expression.

  = equal to.

  != not equal to.

  >= greater than or equal to.

  > greater than

  <= less than or equal to.

  < less than.

- Binary logical operators (cmp_log)

  && all fields must match

  , all fields must match

  || at least one field must match

  # at least one field must match

- Unary logical operators

  ! logical negation

- Grouping operators

  (left parenthesis

) right parenthesis

  [list start

  ] list end

  { list subset start

  } list subset end

Field types and selection operands

Field type restricts the set of operators and values that you may use with the field when defining selection criteria. You can see field type for each field if you run **<lvm command> −S help** where you can find the type name enclosed in square brackets. Currently, LVM recognizes these field types in reports:

- **string** for set of characters (for each string field type, you can use either string or regular expression − regex for the value used in selection criteria)
- **string list** for set of strings
- **number** for integer value
- **size** for integer or floating point number with size unit suffix (see also **lvcreate**(8) man page and description for "−L−−size" option for the list of recognized suffixes)

- **percent** for floating point number with or without "%" suffix (e.g. 50 or 50%)
- **time** for time values

When using **string list** in selection criteria, there are several ways how LVM can match string list fields from report, depending on what list grouping operator is used and what item separator is used within that set of items. Also, note that order of items does not matter here.

- **matching the set strictly** where all items must match – use [ ], e.g. ["a","b","c"]
- **matching a subset of the set** – use { } with "," or "&&" as item delimiter, e.g. {"a","b","c"}
- **matching an intersection with the set** – use { } with "#" or "||" as item delimiter, e.g. {"a" || "b" || "c"}

When using **time** in your selection criteria, LVM can recognize various time formats using standard, absolute or freeform expressions. For examples demonstrating time expressions in selection criteria, see **EXAMPLES** section.

- **Standard time format**

  – date

    YYYY-MM–DD

    YYYY-MM, auto DD=1

    YYYY, auto MM=01 and DD=01

  – time

    hh:mm:ss

    hh:mm, auto ss=0

    hh, auto mm=0, auto ss=0

  – timezone

    +hh:mm or –hh:mm

    +hh or –hh

  The full date/time specification is YYYY-MM–DD hh:mm:ss. Users are able to leave date/time parts from right to left. Whenever these parts are left out, a range is assumed automatically with second granularity. For example:

    "2015-07-07 9:51" means range of "2015-07-07 9:51:00" - "2015-07-07 9:51:59".

    "2015–07" means range of "2015-07-01 0:00:00" - "2015-07-31 23:59:59"

    "2015" means range of "2015-01-01 0:00:00" - "2015-12-31 23:59:59"

- **Absolute time format**

  Absolute time is defined as number of seconds since the Epoch (1970:01:01 00:00 +00:00).

  - @seconds

- **Freeform time format**
  - weekday names ("Sunday" - "Saturday" or abbreviated as "Sun" - "Sat")
  - labels for points in time ("noon", "midnight")
  - labels for a day relative to current day ("today", "yesterday")
  - points back in time with relative offset from today (N is a number)

    "N" "seconds" / "minutes" / "hours" / "days" / "weeks" / "years" "ago"

    "N" "secs" / "mins" / "hrs" ... "ago"

    "N" "s" / "m" / "h" ... "ago"
  - time specification either in hh:mm:ss format or with AM/PM suffixes
  - month names ("January" - "December" or abbreviated as "Jan" - "Dec")

Informal grammar specification

STATEMENT = **column** cmp_op **VALUE** | **STATEMENT** log_op **STATEMENT** | **(STATEMENT)** | **!(STATEMENT)**

VALUE = [VALUE log_op **VALUE]**
For list-based types: string list. Matches strictly. The log_op must always be of one type within the whole list value.

VALUE = {VALUE log_op **VALUE}**
For list-based types: string list. Matches a subset. The log_op must always be of one type within the whole list value.

VALUE = value
For scalar types: number, size, percent, string (or string regex).

EXAMPLES

Basic usage

We start our examples with default configuration - **lvmconfig**(8) is helpful command to display configuration settings which are currently used, including all configuration related to reporting. We will use it throughout examples below to display current configuration.

```
# lvmconfig −−type full global/units global/suffix \
  report/output_format  report/compact_output \
  report/compact_output_cols report/aligned \
  report/headings report/separator \
  report/list_item_separator report/prefixes \
  report/quoted report/columns_as_rows \
  report/binary_values_as_numeric report/time_format \
  report/mark_hidden_devices report/two_word_unknown_device \
  report/buffered
units="h"
suffix=1
output_format="basic"
compact_output=0
compact_output_cols=""
aligned=1
headings=1
```

```
separator=" "
list_item_separator=","
prefixes=0
quoted=1
columns_as_rows=0
binary_values_as_numeric=0
time_format="%Y-%m-%d %T %z"
mark_hidden_devices=1
two_word_unknown_device=0
buffered=1
```

Also, we start with simple LVM layout with two PVs (/dev/sda, /dev/sdb), VG (vg) and two LVs (lvol0 and lvol1) in the VG. We display all possible reports as single commands here, see also **pvs**(8), **vgs**(8), **lvs**(8) man pages for more information. The field set for each report type is configured with configuration settings as we already mentioned in **main report specifics** section in this man page.

```
# lvmconfig −−type full report/pvs_cols report/pvs_sort \
  report/pvsegs_cols report/pvsegs_sort report/vgs_cols \
  report/vgs_sort report/lvs_cols report/lvs_sort \
  report/segs_cols report/segs_sort
pvs_cols="pv_name,vg_name,pv_fmt,pv_attr,pv_size,pv_free"
pvs_sort="pv_name"
pvsegs_cols="pv_name,vg_name,pv_fmt,pv_attr,pv_size,pv_free,
        pvseg_start,pvseg_size"
pvsegs_sort="pv_name,pvseg_start"
vgs_cols="vg_name,pv_count,lv_count,snap_count,vg_attr,vg_size,vg_free"
vgs_sort="vg_name"
lvs_cols="lv_name,vg_name,lv_attr,lv_size,pool_lv,origin,move_pv,
      mirror_log,copy_percent,convert_lv"
lvs_sort="vg_name,lv_name"
segs_cols="lv_name,vg_name,lv_attr,stripes,segtype,seg_size"
segs_sort="vg_name,lv_name,seg_start"

# pvs
  PV         VG Fmt  Attr PSize   PFree
  /dev/sda   vg lvm2 a−−  100.00m 88.00m
  /dev/sdb   vg lvm2 a−−  100.00m 92.00m

# pvs −−segments
  PV         VG Fmt  Attr PSize   PFree  Start SSize
  /dev/sda   vg lvm2 a−−  100.00m 88.00m     0     1
  /dev/sda   vg lvm2 a−−  100.00m 88.00m     1     1
  /dev/sda   vg lvm2 a−−  100.00m 88.00m     2     1
  /dev/sda   vg lvm2 a−−  100.00m 88.00m     3    22
  /dev/sdb   vg lvm2 a−−  100.00m 92.00m     0     1
  /dev/sdb   vg lvm2 a−−  100.00m 92.00m     1     1
  /dev/sdb   vg lvm2 a−−  100.00m 92.00m     2    23

# vgs
  VG #PV #LV #SN Attr   VSize   VFree
  vg   2   2   0 wz−−n- 200.00m 180.00m

# lvs
  LV    VG Attr       LSize Pool Origin Move Log Cpy%Sync Convert
  lvol0 vg −wi−a−−−−− 4.00m
  lvol1 vg rwi−a-r−−− 4.00m                      100.00

# lvs −−segments
  LV    VG Attr       #Str Type   SSize
  lvol0 vg −wi−a−−−−−    1 linear 4.00m
  lvol1 vg rwi−a-r−−−    2 raid1  4.00m
```

We will use **report/lvs_cols** and **report/lvs_sort** configuration settings to define our own list of fields to use and to sort by that is different from defaults. You can do this for other reports in same manner with **report/{pvs,pvseg,vgs,seg}_{cols,sort}** configuration settings. Also note that in the example below, we don't display the "lv_time" field even though we're using it for sorting − this is allowed.

```
# lvmconfig −−type full report/lvs_cols report/lvs_sort
lvs_cols="lv_name,lv_size,origin,pool_lv,copy_percent"
lvs_sort="−lv_time"

# lvs
  LV    LSize Origin Pool Cpy%Sync
  lvol1 4.00m             100.00
  lvol0 4.00m
```

You can use **−o−−options** command line option to override current configuration directly on command line.

```
# lvs −o lv_name,lv_size
  LV    LSize
  lvol1 4.00m
  lvol0 4.00m

# lvs −o+lv_layout
  LV    LSize Origin Pool Cpy%Sync Layout
  lvol1 4.00m             100.00   raid,raid1
  lvol0 4.00m                      linear

# lvs −o-origin
  LV    LSize Pool Cpy%Sync
  lvol1 4.00m      100.00
  lvol0 4.00m

# lvs −o lv_name,lv_size,origin −o+lv_layout −o-origin −O lv_name
  LV    LSize Layout
  lvol0 4.00m linear
  lvol1 4.00m raid,raid1
```

You can obtain the same information with single command where all the information about PVs, PV segments, LVs and LV segments are obtained per VG under a single VG lock for consistency, see also **lvmfullreport**(8) man page for more information. The fullreport has its own configuration settings to define field sets to use, similar to individual reports as displayed above, but configuration settings have "_full" suffix now. This way, it's possible to configure different sets of fields to display and to sort by for individual reports as well as the full report.

```
# lvmconfig −−type full report/pvs_cols_full \
   report/pvs_sort_full report/pvsegs_cols_full \
   report/pvsegs_sort_full report/vgs_cols_full \
  report/vgs_sort_full report/lvs_cols_full \
  report/lvs_sort_full report/segs_cols_full \
  report/segs_sort_full
pvs_cols_full="pv_name,vg_name"
pvs_sort_full="pv_name"
pvsegs_cols_full="pv_name,pvseg_start,pvseg_size"
pvsegs_sort_full="pv_uuid,pvseg_start"
vgs_cols_full="vg_name"
vgs_sort_full="vg_name"
lvs_cols_full="lv_name,vg_name"
lvs_sort_full="vg_name,lv_name"
segs_cols_full="lv_name,seg_start,seg_size"
segs_sort_full="lv_uuid,seg_start"

# lvm fullreport
 VG
 vg
 PV       VG
 /dev/sda vg
 /dev/sdb vg
 LV    VG
 lvol0 vg
 lvol1 vg
 PV       Start SSize
 /dev/sda     0     1
 /dev/sda     1     1
 /dev/sda     2     1
 /dev/sda     3    22
 /dev/sdb     0     1
 /dev/sdb     1     1
 /dev/sdb     2    23
 LV    Start SSize
 lvol0     0 4.00m
 lvol1     0 4.00m
```

Automatic output compaction

If you look at the lvs output above, you can see that the report also contains fields for which there is no information to display (e.g. the columns under "Origin" and "Pool" heading – the "origin" and "pool_lv" fields). LVM can automatically compact report output so such fields are not included in final output. To enable this feature and to compact all fields, use **report/compact_output=1** in your configuration.

```
# lvmconfig --type full report/compact_output
compact_output=1

# lvs
 LV    LSize Cpy%Sync
 lvol1 4.00m 100.00
 lvol0 4.00m

# lvs vg/lvol0
 LV    LSize
 lvol0 4.00m
```

Alternatively, you can define which fields should be compacted by configuring

report/compact_output_cols configuration setting (or **−o−−options #** command line option).

```
# lvmconfig −−type full report/compact_output report/compact_output_cols
compact_output=0
compact_output_cols="origin"

# lvs
  LV    LSize Pool Cpy%Sync
  lvol1 4.00m      100.00
  lvol0 4.00m

# lvs vg/lvol0
  LV    LSize Pool
  lvol0 4.00m

# lvs −o#pool_lv
  LV    LSize Origin Cpy%Sync
  lvol1 4.00m        100.00
  lvol0 4.00m
```

We will use **report/compact_output=1** for subsequent examples.

Further formatting options

By default, LVM displays sizes in reports in human-readable form which means that the most suitable unit is used so it's easy to read. You can use **report/units** configuration setting (or **−−units** option directly on command line) and **report/suffix** configuration setting (or **−−nosuffix** command line option) to change this.

```
# lvs −−units b −−nosuffix
  LV    LSize   Cpy%Sync
  lvol1 4194304 100.00
  lvol0 4194304
```

If you want to configure whether report headings are displayed or not, use **report/headings** configuration settings (or **−−noheadings** command line option).

```
# lvs −−noheadings
  lvol1 4.00m 100.00
  lvol0 4.00m
```

In some cases, it may be useful to display report content as key=value pairs where key here is actually the field name. Use **report/prefixes** configuration setting (or **−−nameprefixes** command line option) to switch between standard output and the key=value output. The key=value pair is the output that is suitable for use in scripts and for other tools to parse easily. Usually, you also don't want to display headings with the output that has these key=value pairs.

```
# lvs −−noheadings −−nameprefixes
  LVM2_LV_NAME='lvol1' LVM2_LV_SIZE='4.00m' LVM2_COPY_PERCENT='100.00'
  LVM2_LV_NAME='lvol0' LVM2_LV_SIZE='4.00m' LVM2_COPY_PERCENT=''
```

To define whether quotation marks in key=value pairs should be used or not, use **report/quoted** configuration setting (or **−−unquoted** command line option).

```
# lvs −−noheadings −−nameprefixes −−unquoted
  LVM2_LV_NAME=lvol1 LVM2_LV_SIZE=4.00m LVM2_COPY_PERCENT=100.00
  LVM2_LV_NAME=lvol0 LVM2_LV_SIZE=4.00m LVM2_COPY_PERCENT=
```

For easier parsing, you can even transpose the report so each column now becomes a row in the output. This is done with **report/output_as_rows** configuration setting (or **−−rows** command line option).

```
# lvs −−noheadings −−nameprefixes −−unquoted −−rows
  LVM2_LV_NAME=lvol1 LVM2_LV_NAME=lvol0
  LVM2_LV_SIZE=4.00m LVM2_LV_SIZE=4.00m
  LVM2_COPY_PERCENT=100.00 LVM2_COPY_PERCENT=
```

Use **report/separator** configuration setting (or **−−separator** command line option) to define your own field separator to use.

```
# lvs −−noheadings −−nameprefixes −−unquoted −−separator " | "
  LVM2_LV_NAME=lvol1 | LVM2_LV_SIZE=4.00m | LVM2_COPY_PERCENT=100.00
  LVM2_LV_NAME=lvol0 | LVM2_LV_SIZE=4.00m | LVM2_COPY_PERCENT=
```

If you are using your own separator, the columns in the output are not aligned by default. Use **report/aligned** configuration setting (or **−−aligned** command line option) for LVM to add extra spaces in report to align the output properly.

```
# lvs −−separator " | "
  LV | LSize | Cpy%Sync
  lvol1 | 4.00m | 100.00
  lvol0 | 4.00m |

# lvs −−separator " | " −−aligned
  LV    | LSize | Cpy%Sync
  lvol1 | 4.00m | 100.00
  lvol0 | 4.00m |
```

Let’s display one one more field in addition ("lv_tags" in this example) for the lvs report output.

```
# lvs −o+lv_tags
  LV    LSize Cpy%Sync LV Tags
  lvol1 4.00m 100.00
  lvol0 4.00m          tagA,tagB
```

The "LV Tags" column in the example above displays two list values, separated by "," character for LV lvol0. If you need different list item separator, use **report/list_item_separator** configuration setting its definition.

```
# lvmconfig −−type full report/list_item_separator
list_item_separator=";"

# lvs −o+tags
  LV    LSize Cpy%Sync LV Tags
  lvol1 4.00m 100.00
  lvol0 4.00m          tagA;tagB
```

But let’s still use the original "," character for list_item_separator for subsequent examples.

Format for any of time values displayed in reports can be configured with **report/time_format** configuretion setting. By default complete date and time is displayed, including timezone.

```
# lvmconfig −−type full report/time_format
time_format="%Y-%m-%d %T %z"

# lvs −o+time
  LV    LSize Cpy%Sync CTime
  lvol1 4.00m 100.00   2016-08-29 12:53:36 +0200
  lvol0 4.00m          2016-08-29 10:15:17 +0200
```

We can change time format in similar way as we do when using **date**(1) command or **strftime**(3) function (**lvmconfig −−type default −−withcomments report/time_format** will give you complete list of available formatting options). In the example below, we decided to use %s for number of seconds since Epoch (1970-01-01 UTC).

```
# lvmconfig −−type full report/time_format
time_format="%s"

# lvs
  LV    Attr       LSize Cpy%Sync LV Tags   CTime
  lvol1 rwi−a-r−−− 4.00m 100.00             1472468016
  lvol0 −wi−a−−−−− 4.00m          tagA,tagB 1472458517
```

The **lvs** does not display hidden LVs by default − to include these LVs in the output, you need to use **−a−−all** command line option. Names for these hidden LVs are displayed within square brackets.

```
# lvs −a
  LV              LSize Cpy%Sync
  lvol1           4.00m 100.00
  [lvol1_rimage_0] 4.00m
  [lvol1_rmeta_0]  4.00m
  [lvol1_rimage_1] 4.00m
  [lvol1_rmeta_1]  4.00m
  lvol0           4.00m
```

You can configure LVM to display the square brackets for hidden LVs or not with **report/mark_hidden_devices** configuration setting.

```
# lvmconfig −−type full report/mark_hidden_devices
mark_hidden_devices=0

# lvs −a
  LV             LSize Cpy%Sync
  lvol1          4.00m 100.00
  lvol1_rimage_0 4.00m
  lvol1_rmeta_0  4.00m
  lvol1_rimage_1 4.00m
  lvol1_rmeta_1  4.00m
  lvol0          4.00m
```

It's not recommended to use LV marks for hidden devices to decide whether the LV is the one to use by end users or not. Please, use "lv_role" field instead which can report whether the LV is "public" or "private". The private LVs are used by LVM only and they should not be accessed directly by end users.

```
# lvs −a −o+lv_role
  LV             LSize Cpy%Sync Role
```

```
 lvol1          4.00m 100.00   public
 lvol1_rimage_0 4.00m          private,raid,image
 lvol1_rmeta_0  4.00m          private,raid,metadata
 lvol1_rimage_1 4.00m          private,raid,image
 lvol1_rmeta_1  4.00m          private,raid,metadata
 lvol0          4.00m          public
```

Some of the reporting fields that LVM reports are of binary nature. For such fields, it's either possible to display word representation of the value (this is used by default) or numeric value (0/1 or −1 in case the value is undefined).

```
# lvs −o+lv_active_locally
 LV    LSize Cpy%Sync ActLocal
 lvol1 4.00m 100.00   active locally
 lvol0 4.00m          active locally
```

We can change the way how these binary values are displayed with **report/binary_values_as_numeric** configuration setting.

```
# lvmconfig −−type full report/binary_values_as_numeric
binary_values_as_numeric=1

# lvs −o+lv_active_locally
 LV    LSize Cpy%Sync ActLocal
 lvol1 4.00m 100.00          1
 lvol0 4.00m                 1
```

Changing output format

LVM can output reports in different formats − use **report/output_format** configuration setting (or **−−reportformat** command line option) to swith the report output format. Currently, LVM supports **"basic"** (all the examples we used above used this format) and **"JSON"** output format.

```
# lvs −o lv_name,lv_size −−reportformat json
 {
    "report": [
       {
          "lv": [
             {"lv_name":"lvol1", "lv_size":"4.00m"},
             {"lv_name":"lvol0", "lv_size":"4.00m"}
          ]
       }
    ]
 }
```

Note that some configuration settings and command line options have no effect with certain report formats. For example, with **JSON** output, it doesn't have any meaning to use **report/aligned** (**−−aligned**), **report/noheadings** (**−−noheadings**), **report/columns_as_rows** (**−−rows**) or **report/buffered** (**−−unbuffered**). All these configuration settings and command line options are ignored if using the **JSON** report output format.

Selection

If you need to select only specific rows from report, you can use LVM's report selection feature. If you call **<lvm_command> −S help**, you'll get quick help on selection. The help contains list of all fields that LVM can use in reports together with its type enclosed in square brackets. The example below contains a line from lvs −S help.

```
# lvs −S help
   ...
   lv_size              - Size of LV in current units. [size]
   ...
```

This line tells you you that the "lv_size" field is of "size" type. If you look at the bottom of the help output, you can see section about "Selection operators" and its "Comparison operators".

```
# lvs −S help
 ...
Selection operators
−−-----------------
Comparison operators:
   =~  - Matching regular expression. [regex]
   !~  - Not matching regular expression. [regex]
    =  - Equal to. [number, size, percent, string, string list, time]
   !=  - Not equal to. [number, size, percent, string, string_list, time]
   >=  - Greater than or equal to. [number, size, percent, time]
    >  - Greater than. [number, size, percent, time]
   <=  - Less than or equal to. [number, size, percent, time]
    <  - Less than. [number, size, percent, time]
since  - Since specified time (same as ’>=’). [time]
after  - After specified time (same as ’>’). [time]
until  - Until specified time (same as ’<=’). [time]
before  - Before specified time (same as ’<’). [time]
 ...
```

Here you can match comparison operators that you may use with the "lv_size" field which is of type "size" - it’s =, !=, >=, >, <= and <. You can find applicable comparison operators for other fields and other field types the same way.

To demostrate selection functionality in LVM, we will create more LVs in addition to lvol0 and lvol1 we used in our previous examples.

```
# lvs −o name,size,origin,snap_percent,tags,time
  LV    LSize Origin Snap%  LV Tags        CTime
  lvol4 4.00m lvol2  24.61                 2016-09-09 16:57:44 +0200
  lvol3 4.00m lvol2  5.08                  2016-09-09 16:56:48 +0200
  lvol2 8.00m              tagA,tagC,tagD 2016-09-09 16:55:12 +0200
  lvol1 4.00m                              2016-08-29 12:53:36 +0200
  lvol0 4.00m              tagA,tagB       2016-08-29 10:15:17 +0200
```

When selecting size and percent fields, we don’t need to use units. For sizes, default "m" (for MiB) is used − this is the same behaviour as already used for LVM commands when specifying sizes (e.g. lvcreate −L). For percent fields, "%" is assumed automatically if it’s not specified. The example below also demonstrates how several criteria can be combined together.

```
# lvs −o name,size,snap_percent −S ’size=8m’
  LV    LSize
  lvol2 8.00m
```

```
# lvs −o name,size,snap_percent −S ’size=8’
```

```
  LV    LSize
  lvol2 8.00m

# lvs −o name,size,snap_percent −S ’size < 5000k’
  LV    LSize Snap%
  lvol4 4.00m 24.61
  lvol3 4.00m 5.08
  lvol1 4.00m
  lvol0 4.00m

# lvs −o name,size,snap_percent −S ’size < 5000k && snap_percent > 20’
  LV    LSize Snap%
  lvol4 4.00m 24.61

# lvs −o name,size,snap_percent \
    −S ’(size < 5000k && snap_percent > 20%) || name=lvol2’
  LV    LSize Snap%
  lvol4 4.00m 24.61
  lvol2 8.00m
```

You can also use selection together with processing-oriented commands.

```
# lvchange −−addtag test −S ’size < 5000k’
  Logical volume vg/lvol1 changed.
  Logical volume vg/lvol0 changed.
  Logical volume vg/lvol3 changed.
  Logical volume vg/lvol4 changed.

# lvchange −−deltag test −S ’tags = test’
  Logical volume vg/lvol1 changed.
  Logical volume vg/lvol0 changed.
  Logical volume vg/lvol3 changed.
  Logical volume vg/lvol4 changed.
```

LVM can recognize more complex values used in selection criteria for string list and time field types. For string lists, you can match whole list strictly, its subset or intersection. Let’s take "lv_tags" field as an example − we select only rows which contain "tagA" within tags field. We’re using { } to denote that we’re interested in subset that matches. If the subset has only one item, we can leave out { }.

```
# lvs −o name,tags −S ’tags={tagA}’
  LV    LV Tags
  lvol2 tagA,tagC,tagD
  lvol0 tagA,tagB

# lvs −o name,tags −S ’tags=tagA’
  LV    LV Tags
  lvol2 tagA,tagC,tagD
  lvol0 tagA,tagB
```

Depending on whether we use "&&" (or ",") or "||" (or "#") as delimiter for items in the set we define in selection criterion for string list, we either match subset ("&&" or ",") or even intersection ("||" or "#").

```
# lvs −o name,tags −S ’tags={tagA,tagC,tagD}’
  LV    LV Tags
```

```
 lvol2 tagA,tagC,tagD

# lvs –o name,tags –S ’tags={tagA || tagC || tagD}’
 LV    LV Tags
 lvol2 tagA,tagC,tagD
 lvol0 tagA,tagB
```

To match the complete set, use [ ] with "&&" (or ",") as delimiter for items. Also note that the order in which we define items in the set is not relevant.

```
# lvs –o name,tags –S ’tags=[tagA]’

# lvs –o name,tags –S ’tags=[tagB,tagA]’
 LV    LV Tags
 lvol0 tagA,tagB
```

If you use [ ] with "||" (or "#"), this is exactly the same as using { }.

```
# lvs –o name,tags –S ’tags=[tagA || tagC || tagD]’
 LV    LV Tags
 lvol2 tagA,tagC,tagD
 lvol0 tagA,tagB
```

To match a set with no items, use "" to denote this (note that we have output compaction enabled so the "LV Tags" column is not displayed in the example below because it’s blank and so it gets compacted).

```
# lvs –o name,tags –S ’tags=""’
 LV
 lvol4
 lvol3
 lvol1

# lvs –o name,tags –S ’tags!=""’
 LV    LV Tags
 lvol2 tagA,tagC,tagD
 lvol0 tagA,tagB
```

When doing selection based on time fields, we can use either standard, absolute or freeform time expressions in selection criteria. Examples below are using standard forms.

```
# lvs –o name,time
 LV    CTime
 lvol4 2016-09-09 16:57:44 +0200
 lvol3 2016-09-09 16:56:48 +0200
 lvol2 2016-09-09 16:55:12 +0200
 lvol1 2016-08-29 12:53:36 +0200
 lvol0 2016-08-29 10:15:17 +0200

# lvs –o name,time –S ’time since "2016-09-01"’
 LV    CTime
 lvol4 2016-09-09 16:57:44 +0200
 lvol3 2016-09-09 16:56:48 +0200
 lvol2 2016-09-09 16:55:12 +0200
```

```
# lvs –o name,time –S 'time since "2016-09-09 16:56"'
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200

# lvs –o name,time –S 'time since "2016-09-09 16:57:30"'
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200

# lvs –o name,time \
    –S 'time since "2016-08-29" && time until "2016-09-09 16:55:12"'
  LV    CTime
  lvol2 2016-09-09 16:55:12 +0200
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200

# lvs –o name,time \
    –S 'time since "2016-08-29" && time before "2016-09-09 16:55:12"'
  LV    CTime
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200
```

Time operators have synonyms: ">=" for since, "<=" for until, ">" for "after" and "<" for "before".

```
# lvs –o name,time \
    –S 'time >= "2016-08-29" && time <= "2016-09-09 16:55:30"'
  LV    CTime
  lvol2 2016-09-09 16:55:12 +0200
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200

# lvs –o name,time \
    –S 'time since "2016-08-29" && time < "2016-09-09 16:55:12"'
  LV    CTime
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200
```

Example below demonstrates using absolute time expression.

```
# lvs –o name,time ––config report/time_format="%s"
  LV    CTime
  lvol4 1473433064
  lvol3 1473433008
  lvol2 1473432912
  lvol1 1472468016
  lvol0 1472458517

# lvs –o name,time –S 'time since @1473433008'
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200
```

Examples below demonstrates using freeform time expressions.

```
# lvs −o name,time −S ’time since "2 weeks ago"’
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200
  lvol2 2016-09-09 16:55:12 +0200
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200

# lvs −o name,time −S ’time since "1 week ago"’
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200
  lvol2 2016-09-09 16:55:12 +0200

# lvs −o name,time −S ’time since "2 weeks ago"’
  LV    CTime
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200

# lvs −o name,time −S ’time before "1 week ago"’
  LV    CTime
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200

# lvs −o name,time −S ’time since "68 hours ago"’
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200
  lvol2 2016-09-09 16:55:12 +0200

# lvs −o name,time −S ’time since "1 year 3 months ago"’
  LV    CTime
  lvol4 2016-09-09 16:57:44 +0200
  lvol3 2016-09-09 16:56:48 +0200
  lvol2 2016-09-09 16:55:12 +0200
  lvol1 2016-08-29 12:53:36 +0200
  lvol0 2016-08-29 10:15:17 +0200
```

Command log reporting

As described in **categorization based on reporting facility** section at the beginning of this document, both **report-oriented** and **processing-oriented** LVM commands can report the command log if this is enabled with **log/report_command_log** configuration setting. Just like any other report, we can set the set of fields to display (**log/command_log_cols**) and to sort by (**log/command_log_sort**) for this report.

```
# lvmconfig −−type full log/report_command_log log/command_log_cols \
   log/command_log_sort log/command_log_selection
report_command_log=1
command_log_cols="log_seq_num,log_type,log_context,log_object_type,
                 log_object_name,log_object_group,log_message,
                 log_errno,log_ret_code"
command_log_sort="log_seq_num"
command_log_selection="!(log_type=status && message=success)"
```

```
# lvs
  Logical Volume
  ==============
  LV    LSize Cpy%Sync
  lvol1 4.00m 100.00
  lvol0 4.00m

  Command Log
  ===========
  Seq LogType Context ObjType ObjName ObjGrp  Msg     Errno RetCode
```

As you can see, the command log is empty (it contains only field names). By default, LVM uses selection on the command log report and this case no row matched the selection criteria, see also **log report specifics** section in this document for more information. We're displaying complete log report in the example below where we can see that both LVs lvol0 and lvol1 were successfully processed as well as the VG vg they are part of.

```
# lvmconfig --type full log/command_log_selection
command_log_selection="all"

# lvs
  Logical Volume
  ==============
  LV    LSize Cpy%Sync
  lvol1 4.00m 100.00
  lvol0 4.00m

  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1
    3 status  processing vg      vg              success     0       1

# lvchange -an vg/lvol1
  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol1   vg      success     0       1
    2 status  processing vg      vg              success     0       1
```

Handling multiple reports per single command

To configure the log report directly on command line, we need to use **--configreport** option before we start any **-o--options**, **-O--sort** or **-S--select** that is targeted for log report.

```
# lvs -o lv_name,lv_size --configreport log -o log_object_type, \
   log_object_name,log_message,log_ret_code
  Logical Volume
  ==============
  LV    LSize
  lvol1 4.00m
  lvol0 4.00m

  Command Log
```

```
 ===========
 ObjType ObjName Msg     RetCode
 lv      lvol0   success       1
 lv      lvol1   success       1
 vg      vg      success       1
```

The **lvm fullreport**, with or without log report, consists of several reports – the **−−configreport** is also used to target particular subreport here.

Below is an extended example with **lvm fullreport** to illustrate combination of various options. The report output is in JSON format. Also, we configure "vg", "pvseg", "seg" and "log" subreport to contain only specified fields. For the "pvseg" subreport, we're intested only in PV names having "sda" in their name. For the "log" subreport we're intested only in log lines related to either "lvol0" object or object having "sda" in its name. Also, for the log subreport we define ordering to be based on "log_object_type" field.

```
# lvm fullreport −−reportformat json \
  −−configreport vg −o vg_name,vg_size \
  −−configreport pvseg −o pv_name,pvseg_start \
                 −S 'pv_name=˜sda' \
  −−configreport seg −o lv_name,seg_start \
  −−configreport log −o log_object_type,log_object_name \
               −O log_object_type \
               −S 'log_object_name=lvol0 || \
                  log_object_name=˜sda'
 {
    "report": [
        {
           "vg": [
              {"vg_name":"vg", "vg_size":"200.00m"}
           ]
           ,
           "pv": [
              {"pv_name":"/dev/sda", "vg_name":"vg"},
              {"pv_name":"/dev/sdb", "vg_name":"vg"}
           ]
           ,
           "lv": [
              {"lv_name":"lvol0", "vg_name":"vg"},
              {"lv_name":"lvol1", "vg_name":"vg"}
           ]
           ,
           "pvseg": [
              {"pv_name":"/dev/sda", "pvseg_start":"0"},
              {"pv_name":"/dev/sda", "pvseg_start":"1"},
              {"pv_name":"/dev/sda", "pvseg_start":"2"},
              {"pv_name":"/dev/sda", "pvseg_start":"3"}
           ]
           ,
           "seg": [
              {"lv_name":"lvol0", "seg_start":"0 "},
              {"lv_name":"lvol1", "seg_start":"0 "}
           ]
        }
    ]
```

```
      ,
      "log": [
        {"log_object_type":"lv", "log_object_name":"lvol0"},
        {"log_object_type":"lv", "log_object_name":"lvol0"},
        {"log_object_type":"pv", "log_object_name":"/dev/sda"},
        {"log_object_type":"pv", "log_object_name":"/dev/sda"},
      ]
    }
```

Report extensions for LVM shell

As already stated in **log report coverage** paragraph under **log report specifics** in this documentation, when using **LVM shell** the **log report** coverage is wider. There's also special command designed to query last command's log report in the **LVM shell** - the **lastlog** command.

The example below illustrates a situation where we called lvs command. After that, we inspected the log report with the **lastlog**, without any selection so all the log report is displayed on output. Then we called **lastlog** further, giving various selection criteria. Then we ran unknown LVM command "abc" for which the log report displays appropriate failure state.

```
# lvm
lvm> lvs
  Logical Volume
  ==============
  LV    LSize Cpy%Sync
  lvol1 4.00m 100.00
  lvol0 4.00m

  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1
    3 status  processing vg      vg              success     0       1
    4 status  shell      cmd     lvs             success     0       1

lvm> lastlog
  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1
    3 status  processing vg      vg              success     0       1
    4 status  shell      cmd     lvs             success     0       1

lvm> lastlog −S log_object_type=lv
  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1

lvm> lastlog −S log_context=shell
  Command Log
  ===========
  Seq LogType Context ObjType ObjName ObjGrp  Msg     Errno RetCode
    4 status  shell   cmd     lvs             success     0       1

lvm> abc
  Command Log
  ===========
  Seq LogType Context ObjType ObjName ObjGrp  Msg                              Errno RetCode
    1 error   shell   cmd     abc             No such command 'abc'.  Try 'help'.    -1       0
    2 status  shell   cmd     abc             failure                          -1       2
```

SEE ALSO

lvm (8), **lvmconfig** (8), **lvm fullreport** (8)