



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

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'multipath.conf.5'***

***\$ man multipath.conf.5***

MULTIPATH.CONF(5)      File Formats Manual      MULTIPATH.CONF(5)

NAME

    multipath.conf - multipath daemon configuration file.

DESCRIPTION

    /etc/multipath.conf is the configuration file for the multipath daemon.

    It is used to overwrite the built-in configuration table of multipathd.

    Any line whose first non-white-space character is a '#' is considered a comment line. Empty lines are ignored.

    Currently used multipathd configuration can be displayed with the `mul?tipath -t` or `multipathd show config` command.

SYNTAX

    The configuration file contains entries of the form:

```
<section> {
    <attribute> <value>
    ...
    <subsection> {
        <attribute> <value>
        ...
```

```
}
```

```
}
```

Each section contains one or more attributes or subsections. The recognized keywords for attributes or subsections depend on the section in which they occur.

<attribute> and <value> must be on a single line. <attribute> is one of the keywords listed in this man page. <value> is either a simple word (containing no whitespace and none of the characters '"', '#', and '!') or one string enclosed in double quotes ("..."). Outside a quoted string, text starting with '#', and '!' is regarded as a comment and ignored until the end of the line. Inside a quoted string, '#' and '!' are normal characters, and whitespace is preserved. To represent a double quote character inside a double quoted string, use two consecutive double quotes (""). Thus '2.5" SSD' can be written as "2.5" SSD".

Opening braces ('{') must follow the (sub)section name on the same line. Closing braces ('}') that mark the end of a (sub)section must be the only non-whitespace character on the line. Whitespace is ignored except inside double quotes, thus the indentation shown in the above example is helpful for human readers but not mandatory.

Note on regular expressions: The multipath.conf syntax allows many attribute values to be specified as POSIX Extended Regular Expressions (see regex(7)). These regular expressions are case sensitive and not anchored, thus the expression "bar" matches "barbie", "rhabarber", and "wunderbar", but not "Barbie". To avoid unwanted substring matches, standard regular expression syntax using the special characters "^" and "\$" can be used.

The following section keywords are recognized:

**defaults** This section defines default values for attributes which are used whenever no values are given in the appropriate device or multipath sections.

**blacklist** This section defines which devices should be excluded from the multipath topology discovery.

## blacklist\_exceptions

This section defines which devices should be included in the multipath topology discovery, despite being listed in the blacklist section.

**multipaths** This section defines the multipath topologies. They are indexed by a World Wide Identifier(WWID). For details on the WWID generation see section WWID generation below. Attributes set in this section take precedence over all others.

**devices** This section defines the device-specific settings. Devices are identified by vendor, product, and revision.

**overrides** This section defines values for attributes that should override the device-specific settings for all devices.

## defaults section

The defaults section recognizes the following keywords:

**verbosity** Default verbosity. Higher values increase the verbosity level. Valid levels are between 0 and 6.  
The default is: 2

**polling\_interval** Interval between two path checks in seconds. For properly functioning paths, the interval between checks will gradually increase to `max_polling_interval`. This value will be overridden by the `WatchdogSec` setting in the `multipathd.service` definition if `systemd` is used.  
The default is: 5

**max\_polling\_interval**  
Maximal interval between two path checks in seconds.  
The default is:  $4 * \text{polling\_interval}$

**reassign\_maps** Enable reassigning of device-mapper maps. With this option `multipathd` will remap existing device-mapper maps to always point to multipath device, not the underlying block devices. Possible values are `yes` and `no`.  
The default is: `no`

`multipath_dir` This option is deprecated, and will be removed in a future release. Directory where the dynamic shared objects are stored. Defined at compile time, commonly `/lib64/multipath/` or `/lib/multipath/`.

The default is: `<system dependent>`

`path_selector` The default path selector algorithm to use; they are offered by the kernel multipath target:

`round-robin 0`

Loop through every path in the path group, sending the same amount of I/O to each.

Some aspects of behavior can be controlled with the attributes: `rr_min_io`, `rr_min_io_rq` and `rr_weight`.

`queue-length 0`

(Since 2.6.31 kernel) Choose the path for the next bunch of I/O based on the amount of outstanding I/O to the path.

`service-time 0`

(Since 2.6.31 kernel) Choose the path for the next bunch of I/O based on the amount of outstanding I/O to the path and its relative throughput.

`historical-service-time 0`

(Since 5.8 kernel) Choose the path for the next bunch of I/O based on the estimation of future service time based on the history of previous I/O submitted to each path.

The default is: `service-time 0`

`path_grouping_policy`

The default path grouping policy to apply to unspecified multipaths. Possible values are:

`failover` One path per priority group.

multibus All paths in one priority group.

group\_by\_serial

One priority group per serial number.

group\_by\_prio

One priority group per priority value.

Priorities are determined by callout programs specified as a global, per-controller or per-multipath option in the configuration file.

group\_by\_node\_name

One priority group per target node name.

Target node names are fetched in `/sys/class/fc_transport/target*/node_name`.

The default is: failover

`uid_attrs` Setting this option activates merging uevents by WWID, which may improve uevent processing efficiency. Moreover, it's an alternative method to configure the udev properties to use for determining unique path identifiers (WWIDs).

The value of this option is a space separated list of records like "type:ATTR", where type is matched against the beginning of the device node name (e.g. `sd:ATTR` matches `sda`), and ATTR is the name of the udev property to use for matching devices.

If this option is configured and matches the device node name of a device, it overrides any other configured methods for determining the WWID for this device.

The default is: <unset>. To enable uevent merging, set it e.g. to "sd:ID\_SERIAL dasd:ID\_UID nvme:ID\_WWN".

`uid_attribute` The udev attribute providing a unique path identifier (WWID). If `uid_attribute` is set to the empty string, WWID determination is done using the `sysfs` method

rather than using udev (not recommended in production; see WWID generation below).

The default is: ID\_SERIAL, for SCSI devices

The default is: ID\_UID, for DASD devices

The default is: ID\_WWN, for NVMe devices

getuid\_callout (Superseded by uid\_attribute) The default program and

args to callout to obtain a unique path identifier.

Should be specified with an absolute path.

The default is: <unset>

prio The name of the path priority routine. The specified routine should return a numeric value specifying the relative priority of this path. Higher number have a higher priority. "none" is a valid value. Currently the following path priority routines are implemented:

const Return a constant priority of 1.

sysfs Use the sysfs attributes access\_state and preferred\_path to generate the path priority. This prioritizer accepts the optional prio\_arg exclusive\_pref\_bit.

emc (Hardware-dependent) Generate the path priority for DGC class arrays as CLARiiON CX/AX and EMC VNX and Unity families.

alua (Hardware-dependent) Generate the path priority based on the SCSI-3 ALUA settings. This prioritizer accepts the optional prio\_arg exclusive\_pref\_bit.

ontap (Hardware-dependent) Generate the path priority for NetApp ONTAP class and OEM arrays as IBM NSeries.

rdac (Hardware-dependent) Generate the path priority for LSI/Engenio/NetApp RDAC class as NetApp SANtricity E/EF Series, and OEM arrays from IBM DELL SGI STK and SUN.

hp\_sw (Hardware-dependent) Generate the path priority for HP/COMPAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode exclusively.

hds (Hardware-dependent) Generate the path priority for Hitachi AMS families of arrays other than AMS 2000.

random Generate a random priority between 1 and 10.

weightedpath

Generate the path priority based on the regular expression and the priority provided as argument. Requires prio\_args keyword.

path\_latency

Generate the path priority based on a latency algorithm. Requires prio\_args keyword.

ana (Hardware-dependent) Generate the path priority based on the NVMe ANA settings.

datacore (Hardware-dependent) Generate the path priority for some DataCore storage arrays. Requires prio\_args keyword.

iet (iSCSI only) Generate path priority for iSCSI targets based on IP address. Requires prio\_args keyword.

The default depends on the detect\_prio setting: If detect\_prio is yes (default), the default priority algorithm is sysfs (except for NetAPP E-Series, where it is alua). If detect\_prio is no, the default priority algorithm is const.

prio\_args Arguments to pass to the prio function. This only applies to certain prioritizers:

weighted Needs a value of the form "<hctl|dev?

name|serial|wwn> <regex1> <prio1> <regex2>  
<prio2> ..."

hctl Regex can be of SCSI H:B:T:L for?

mat. For example: 1:0:..: ,

\*:0:0:.

devname Regex can be of device name for?

mat. For example: sda , sd.e

serial Regex can be of serial number for?

mat. For example: .\*J1FR.\*324 .

The serial can be looked up

through sysfs or by running multi?

pathd show paths format "%z". For

example: 0395J1FR904324

wwn Regex can be of the form

"host\_wwnn:host\_wwpn:tar?

get\_wwnn:target\_wwpn" these values

can be looked up through sysfs or

by running multipathd show paths

format "%N:%R:%n:%r". For example:

0x200100e08ba0aea0:0x210100e08ba0aea0:.\*.\*

, .\*.\*:iqn.2009-10.com.red?

hat.msp.lab.ask-06:.\*

path\_latency

Needs a value of the form "io\_num=<20>

base\_num=<10>"

io\_num The number of read IOs sent to the

current path continuously, used to

calculate the average path la?

tency. Valid Values: Integer, [2,

200].

base\_num

The base number value of logarith?



mic scale, used to partition different priority ranks. Valid Values: Integer, [2, 10]. And Max average latency value is 100s, min average latency value is 1us. For example: If base\_num=10, the paths will be grouped in priority groups with path latency <=1us, (1us, 10us], (10us, 100us], (100us, 1ms], (1ms, 10ms], (10ms, 100ms], (100ms, 1s], (1s, 10s], (10s, 100s], >100s.

**alua** If `exclusive_pref_bit` is set, paths with the preferred path bit set will always be in their own path group.

**sysfs** If `exclusive_pref_bit` is set, paths with the preferred path bit set will always be in their own path group.

**datacore**

`preferredsds`

(Mandatory) The preferred "SDS name".

`timeout` (Optional) The timeout for the IN?

QUIRY, in ms.

**iet**

`preferredip=...`

(Mandatory) The preferred IP address, in dotted decimal notation, for iSCSI targets.

The default is: <unset>

**features** Specify any device-mapper features to be used. Syntax is `num list` where `num` is the number, between 0 and 8, of features in list. Possible values for the feature

list are:

`queue_if_no_path`

(Deprecated, superseded by `no_path_retry`)

Queue I/O if no path is active. Identical to the `no_path_retry` with `queue` value. If both this feature and `no_path_retry` are set, the latter value takes precedence.

See KNOWN ISSUES.

`pg_init_retries` <times>

(Since kernel 2.6.24) Number of times to retry `pg_init`, it must be between 1 and 50.

`pg_init_delay_msecs` <msecs>

(Since kernel 2.6.38) Number of msecs before `pg_init` retry, it must be between 0 and 60000.

`queue_mode` <mode>

(Since kernel 4.8) Select the queueing mode per multipath device. <mode> can be `bio`, `rq` or `mq`, which corresponds to bio-based, request-based, and block-multiqueue (`blk-mq`) request-based, respectively. Before kernel 4.20 The default depends on the kernel parameter `dm_mod.use_blk_mq`. It is `mq` if the latter is set, and `rq` otherwise. Since kernel 4.20, `rq` and `mq` both correspond to block-multiqueue. Once a multipath device has been created, its `queue_mode` cannot be changed. `nvme:tcp` paths are only supported in multipath devices with `queue_mode` set to `bio`. multipath will automatically set this when creating a device with `nvme:tcp` paths.

The default is: <unset>

`path_checker` The default method used to determine the path's state.

The synchronous checkers (all except `tur` and `directio`) will cause `multipathd` to pause most activity, waiting up to `checker_timeout` seconds for the path to respond.

The asynchronous checkers (`tur` and `directio`) will not pause `multipathd`. Instead, `multipathd` will check for a response once per second, until `checker_timeout` seconds have elapsed. Possible values are:

`readsector0` (Deprecated) Read the first sector of the device. This checker is being deprecated, please use `tur` instead.

`tur` Issue a TEST UNIT READY command to the device.

`emc_clariion`

(Hardware-dependent) Query the DGC/EMC specific EVPD page 0xC0 to determine the path state for CLARiiON CX/AX and EMC VNX and Unity arrays families.

`hp_sw` (Hardware-dependent) Check the path state for HP/COMPAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode exclusively.

`rdac` (Hardware-dependent) Check the path state for LSI/Engenio/NetApp RDAC class as NetApp SANtricity E/EF Series, and OEM arrays from IBM DELL SGI STK and SUN.

`directio` Read the first sector with direct I/O.

This checker could cause spurious path failures under high load. Increasing `checker_timeout` can help with this.

`cciss_tur` (Hardware-dependent) Check the path state for HP/COMPAQ Smart Array(CCISS) controllers.

none Do not check the device, fallback to use the values retrieved from sysfs

The default is: tur

alias\_prefix The user\_friendly\_names prefix.

The default is: mpath

failback Tell multipathd how to manage path group failback. To select immediate or a value, it's mandatory that the device has support for a working prioritizer.

immediate Immediately fallback to the highest priority pathgroup that contains active paths.

manual Do not perform automatic failback.

followover Used to deal with multiple computers accessing the same Active/Passive storage devices. Only perform automatic failback when the first path of a pathgroup becomes active. This keeps a cluster node from automatically failing back when another node requested the failover.

values > 0 Deferred failback (time to defer in seconds).

The default is: manual

rr\_min\_io Number of I/O requests to route to a path before switching to the next in the same path group. This is only for Block I/O(BIO) based multipath and only apply to round-robin path\_selector.

The default is: 1000

rr\_min\_io\_rq Number of I/O requests to route to a path before switching to the next in the same path group. This is only for Request based multipath and only apply to round-robin path\_selector.

The default is: 1

max\_fds Specify the maximum number of file descriptors that can be opened by multipath and multipathd. This is

equivalent to `ulimit -n`. A value of `max` will set this to the system limit from `/proc/sys/fs/nr_open`. If this is not set, the maximum number of open fds is taken from the calling process. It is usually 1024. To be safe, this should be set to the maximum number of paths plus 32, if that number is greater than 1024.

The default is: `max`

`rr_weight` If set to priorities the multipath configurator will assign path weights as "`path prio * rr_min_io`". Possible values are priorities or `uniform`. Only apply to round-robin `path_selector`.

The default is: `uniform`

`no_path_retry` Specify what to do when all paths are down. Possible values are:

`value > 0` Number of retries until disable I/O queueing.

`fail` For immediate failure (no I/O queueing).

`queue` For never stop I/O queueing, similar to `queue_if_no_path`. See KNOWN ISSUES.

The default is: `fail`

`queue_without_daemon`

If set to `no`, when `multipathd` stops, queueing will be turned off for all devices. This is useful for devices that set `no_path_retry`. If a machine is shut down while all paths to a device are down, it is possible to hang waiting for I/O to return from the device after `multipathd` has been stopped. Without `multipathd` running, access to the paths cannot be restored, and the kernel cannot be told to stop queueing I/O. Setting `queue_without_daemon` to `no`, avoids this problem.

The default is: `no`

`checker_timeout` Specify the timeout to use for path checkers and pri?

oritizers, in seconds. Only prioritizers that issue scsi commands use checker\_timeout. If a path does not respond to the checker command after checker\_timeout seconds have elapsed, it is considered down.

The default is: in /sys/block/<dev>/device/timeout

#### allow\_usb\_devices

If set to no, all USB devices will be skipped during path discovery. If you intend to use multipath on USB attached devices, set this to yes.

The default is: no

#### flush\_on\_last\_del

If set to yes, multipathd will disable queueing when the last path to a device has been deleted.

The default is: no

#### user\_friendly\_names

If set to yes, using the bindings file /etc/multipath/bindings to assign a persistent and unique alias to the multipath, in the form of mpath<n>. If set to no use the WWID as the alias. In either case this will be overridden by any specific aliases in the multipaths section.

The default is: no

#### fast\_io\_fail\_tmo

Specify the number of seconds the SCSI layer will wait

after a problem has been detected on a FC remote port before failing I/O to devices on that remote port.

This should be smaller than dev\_loss\_tmo. Setting this to off will disable the timeout.

The default is: 5

#### dev\_loss\_tmo

Specify the number of seconds the SCSI layer will wait

after a problem has been detected on a FC remote port before removing it from the system. This can be set to

"infinity" which sets it to the max value of

2147483647 seconds, or 68 years. It will be automatic?

cally adjusted to the overall retry interval  
no\_path\_retry \* polling\_interval if a number of re?  
tries is given with no\_path\_retry and the overall  
retry interval is longer than the specified  
dev\_loss\_tmo value. The Linux kernel will cap this  
value to 600 if fast\_io\_fail\_tmo is not set. See KNOWN  
ISSUES.

The default is: 600

eh\_deadline Specify the maximum number of seconds the SCSI layer  
will spend doing error handling when scsi devices  
fail. After this timeout the scsi layer will perform a  
full HBA reset. Setting this may be necessary in cases  
where the rport is never lost, so fast\_io\_fail\_tmo and  
dev\_loss\_tmo will never trigger, but (frequently do to  
load) scsi commands still hang. Note: when the scsi  
error handler performs the HBA reset, all target paths  
on that HBA will be affected. eh\_deadline should only  
be set in cases where all targets on the affected HBAs  
are multipathed.

The default is: <unset>

bindings\_file This option is deprecated, and will be removed in a  
future release. The full pathname of the binding file  
to be used when the user\_friendly\_names option is set.

The default is: /etc/multipath/bindings

wwids\_file This option is deprecated, and will be removed in a  
future release. The full pathname of the WWIDs file,  
which is used by multipath to keep track of the WWIDs  
for LUNs it has created multipath devices on in the  
past.

The default is: /etc/multipath/wwids

prkeys\_file This option is deprecated, and will be removed in a  
future release. The full pathname of the prkeys file,  
which is used by multipathd to keep track of the per?

sistent reservation key used for a specific WWID, when reservation\_key is set to file.

The default is: /etc/multipath/prkeys

log\_checker\_err If set to once , multipathd logs the first path checker error at logging level 2. Any later errors are logged at level 3 until the device is restored. If set to always , multipathd always logs the path checker error at logging level 2.

The default is: always

reservation\_key This is the service action reservation key used by mpathpersist. It must be set for all multipath devices using persistent reservations, and it must be the same as the RESERVATION KEY field of the PERSISTENT RESERVE OUT parameter list which contains an 8-byte value provided by the application client to the device server to identify the L\_T nexus. If the --param-aptpl option is used when registering the key with mpathpersist, :aptpl must be appended to the end of the reservation key.

Alternatively, this can be set to file, which will store the RESERVATION KEY registered by mpathpersist in the prkeys\_file. multipathd will then use this key to register additional paths as they appear. When the registration is removed, the RESERVATION KEY is removed from the prkeys\_file. The prkeys file will automatically keep track of whether the key was registered with --param-aptpl.

The default is: <unset>

all\_tg\_pt Set the 'all targets ports' flag when registering keys with mpathpersist. Some arrays automatically set and clear registration keys on all target ports from a host, instead of per target port per host. The ALL\_TG\_PT flag must be set to successfully use mpath?



persist on these arrays. Setting this option is identical to calling `mpathpersist` with `--param-alltgpt`

The default is: no

`retain_attached_hw_handler`

(Obsolete for kernels  $\geq 4.3$ ) If set to yes and the SCSI layer has already attached a `hardware_handler` to the device, multipath will not force the device to use the `hardware_handler` specified by `mutipath.conf`. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler.

The default is: yes

Important Note: Linux kernel 4.3 or newer always behaves as if "retain\_attached\_hw\_handler yes" was set.

`detect_prio` If set to yes, multipath will try to detect if the device supports SCSI-3 ALUA. If so, the device will automatically use the `sysfs` prioritizer if the required `sysf` attributes `access_state` and `preferred_path` are supported, or the `alua` prioritizer if not. If set to no, the prioritizer will be selected as usual.

The default is: yes

`detect_checker` if set to yes, multipath will try to detect if the device supports SCSI-3 ALUA. If so, the device will automatically use the `tur` checker. If set to no, the checker will be selected as usual.

The default is: yes

`force_sync` If set to yes, `multipathd` will call the path checkers in `sync` mode only. This means that only one checker will run at a time. This is useful in the case where many `multipathd` checkers running in parallel causes significant CPU pressure.

The default is: no

`strict_timing` If set to yes, `multipathd` will start a new path

checker loop after exactly one second, so that each path check will occur at exactly `polling_interval` seconds. On busy systems path checks might take longer than one second; here the missing ticks will be accounted for on the next round. A warning will be printed if path checks take longer than `polling_interval` seconds.

The default is: no

`deferred_remove` If set to yes, multipathd will do a deferred remove instead of a regular remove when the last path device has been deleted. This means that if the multipath device is still in use, it will be freed when the last user closes it. If path is added to the multipath device before the last user closes it, the deferred remove will be canceled.

The default is: no

`partition_delimiter`

This parameter controls how multipath chooses the names of partition devices of multipath maps if a multipath map is renamed (e.g. if a map alias is added or changed). If this parameter is set to a string other than `"/UNSET/"` (even the empty string), multipath inserts that string between device name and partition number to construct the partition device name. Otherwise (i.e. if this parameter is unset or has the value `"/UNSET/"`), the behavior depends on the map name: if it ends in a digit, a "p" is inserted between name and partition number; otherwise, the partition number is simply appended. Distributions may use a non-null default value for this option; in this case, the user must set it to `"/UNSET/"` to obtain the original `<unset>` behavior. Use `multipath -T` to check the current settings.

The default is: <unset>

`config_dir` This option is deprecated, and will be removed in a future release. If set to anything other than "", multipath will search this directory alphabetically for file ending in ".conf" and it will read configuration information from them, just as if it was in /etc/multipath.conf. `config_dir` must either be "" or a fully qualified directory name.

The default is: /etc/multipath/conf.d/

`san_path_err_threshold`

If set to a value greater than 0, multipathd will watch paths and check how many times a path has been failed due to errors. If the number of failures on a particular path is greater than the `san_path_err_threshold`, then the path will not reinitialize till `san_path_err_recovery_time`. These path failures should occur within a `san_path_err_forget_rate` checks, if not we will consider the path is good enough to reinitialize. See "Shaky paths detection" below.

The default is: no

`san_path_err_forget_rate`

If set to a value greater than 0, multipathd will check whether the path failures has exceeded the `san_path_err_threshold` within this many checks i.e `san_path_err_forget_rate`. If so we will not reinitialize the path till `san_path_err_recovery_time`. See "Shaky paths detection" below.

The default is: no

`san_path_err_recovery_time`

If set to a value greater than 0, multipathd will make sure that when path failures has exceeded the `san_path_err_threshold` within `san_path_err_forget_rate`

then the path will be placed in failed state for `san_path_err_recovery_time` duration. Once `san_path_err_recovery_time` has timeout we will reinstate the failed path. `san_path_err_recovery_time` value should be in secs. See "Shaky paths detection" below.

The default is: no

#### `marginal_path_double_failed_time`

One of the four parameters of supporting path check based on accounting IO error such as intermittent error. When a path failed event occurs twice in `marginal_path_double_failed_time` seconds due to an IO error and all the other three parameters are set, `multipathd` will fail the path and enqueue this path into a queue of which members are sent a couple of continuous direct reading asynchronous IOs at a fixed sample rate of 10HZ to start IO error accounting process. See "Shaky paths detection" below.

The default is: no

#### `marginal_path_err_sample_time`

One of the four parameters of supporting path check based on accounting IO error such as intermittent error. If it is set to a value no less than 120, when a path fail event occurs twice in `marginal_path_double_failed_time` second due to an IO error, `multipathd` will fail the path and enqueue this path into a queue of which members are sent a couple of continuous direct reading asynchronous IOs at a fixed sample rate of 10HZ to start the IO accounting process for the path will last for `marginal_path_err_sample_time`. If the rate of IO error on a particular path is greater than the `marginal_path_err_rate_threshold`, then the path will not reinstate for `san_path_err_recovery_time`

marginal\_path\_err\_recheck\_gap\_time seconds unless there is only one active path. After marginal\_path\_err\_recheck\_gap\_time expires, the path will be requeued for rechecking. If checking result is good enough, the path will be reinstated. See "Shaky paths detection" below.

The default is: no

marginal\_path\_err\_rate\_threshold

The error rate threshold as a permillage (1/1000). One of the four parameters of supporting path check based on accounting IO error such as intermittent error. Refer to marginal\_path\_err\_sample\_time. If the rate of IO errors on a particular path is greater than this parameter, then the path will not reinstate for marginal\_path\_err\_recheck\_gap\_time seconds unless there is only one active path. See "Shaky paths detection" below.

The default is: no

marginal\_path\_err\_recheck\_gap\_time

One of the four parameters of supporting path check based on accounting IO error such as intermittent error. Refer to marginal\_path\_err\_sample\_time. If this parameter is set to a positive value, the failed path of which the IO error rate is larger than marginal\_path\_err\_rate\_threshold will be kept in failed state for marginal\_path\_err\_recheck\_gap\_time seconds. When marginal\_path\_err\_recheck\_gap\_time seconds expires, the path will be requeued for checking. If checking result is good enough, the path will be reinstated, or else it will keep failed. See "Shaky paths detection" below.

The default is: no

delay\_watch\_checks

This option is deprecated, and mapped to `san_path_err_forget_rate`. If this is set to a value greater than 0 and no `san_path_err` options are set, `san_path_err_forget_rate` will be set to the value of `delay_watch_checks` and `san_path_err_threshold` will be set to 1. See the `san_path_err_forget_rate` and `san_path_err_threshold` options, and "Shaky paths detection" below for more information.

The default is: no

#### `delay_wait_checks`

This option is deprecated, and mapped to `san_path_err_recovery_time`. If this is set to a value greater than 0 and no `san_path_err` options are set, `san_path_err_recovery_time` will be set to the value of `delay_wait_checks` times `max_polling_interval`. This will give approximately the same wait time as `delay_wait_checks` previously did. Also, `san_path_err_threshold` will be set to 1. See the `san_path_err_recovery_time` and `san_path_err_threshold` options, and "Shaky paths detection" below for more information.

The default is: no

#### `marginal_pathgroups`

If set to off, the `delay_*_checks`, `marginal_path_*`, and `san_path_err_*` options will keep marginal, or "shaky", paths from being reinstated until they have been monitored for some time. This can cause situations where all non-marginal paths are down, and no paths are usable until `multipathd` detects this and reinstates a marginal path. If the multipath device is not configured to queue IO in this case, it can cause IO errors to occur, even though there are marginal paths available. However, if this option is set to

on, when one of the marginal path detecting methods determines that a path is marginal, it will be reinserted and placed in a separate pathgroup that will only be used after all the non-marginal pathgroups have been tried first. This prevents the possibility of IO errors occurring while marginal paths are still usable. After the path has been monitored for the configured time, and is declared healthy, it will be returned to its normal pathgroup. If this option is set to `fpin`, `multipathd` will receive `fpin` notifications, set path states to "marginal" accordingly, and regroup paths as described for `on`. This option can't be used in combination with other options for "Shaky path detection" (see below). Note: If this is set to `fpin`, the `marginal_path_*` and `san_path_err_*` options are implicitly set to `no`. Also, this option cannot be switched either to or from `fpin` on a `multipathd` reconfigure. `multipathd` must be restarted for the change to take effect. See "Shaky paths detection" below for more information.

The default is: `off`

`find_multipaths` This option controls whether `multipath` and `multipathd` try to create multipath maps over non-blacklisted devices they encounter. This matters a) when a device is encountered by `multipath -u` during udev rule processing (a device is blocked from further processing by higher layers - such as LVM - if and only if it's considered a valid multipath device path), and b) when `multipathd` detects a new device. The following values are possible:

`strict` Both `multipath` and `multipathd` treat only such devices as multipath devices which have been part of a multipath map previously, and

- which are therefore listed in the `wwids_file`. Users can manually set up multipath maps using the `multipathd add map com?mand`. Once set up manually, the map is remembered in the `wwids` file and will be set up automatically in the future.
- no Multipath behaves like strict. Multipathd behaves like greedy.
- yes Both `multipathd` and `multipath` treat a device as multipath device if the conditions for strict are met, or if at least two non-blacklisted paths with the same WWID have been detected.
- greedy Both `multipathd` and `multipath` treat every non-blacklisted device as multipath device path.
- smart This differs from `find_multipaths` yes only in the way it treats new devices for which only one path has been detected yet. When such a device is first encountered in udev rules, it is treated as a multipath device. `multipathd` waits whether additional paths with the same WWID appears. If that happens, it sets up a multipath map. If it doesn't happen until a timeout expires, or if setting up the map fails, a new uevent is triggered for the device; at second encounter in the udev rules, the device will be treated as non-multipath and passed on to upper layers. Note: this may cause delays during device detection if there are single-path devices which aren't blacklisted.



## find\_multipaths\_timeout

Timeout, in seconds, to wait for additional paths after detecting the first one, if find\_multipaths "smart" (see above) is set. If the value is positive, this timeout is used for all unknown, non-blacklisted devices encountered. If the value is negative (recommended), it's only applied to "known" devices that have an entry in multipath's hardware table, either in the built-in table or in a device section; other ("unknown") devices will use a timeout of only 1 second to avoid booting delays. The value 0 means "use the built-in default". If find\_multipath has a value other than smart, this option has no effect.

The default is: -10 (10s for known and 1s for unknown hardware)

## uxsock\_timeout

CLI receive timeout in milliseconds. For larger systems CLI commands might timeout before the multipathd lock is released and the CLI command can be processed. This will result in errors like "timeout receiving packet" to be returned from CLI commands. In these cases it is recommended to increase the CLI timeout to avoid those issues.

The default is: 4000

## retrigger\_tries

Sets the number of times multipathd will try to retrigger a uevent to get the WWID.

The default is: 3

## retrigger\_delay

Sets the amount of time, in seconds, to wait between retriggers.

The default is: 10

## missing\_uev\_wait\_timeout

Controls how many seconds multipathd will wait, after a new multipath device is created, to receive a change event from udev for the device, before automatically

enabling device reloads. Usually multipathd will delay reloads on a device until it receives a change uevent from the initial table load.

The default is: 30

`skip_kpartx` If set to yes, kpartx will not automatically create partitions on the device.

The default is: no

`disable_changed_wwids`

This option is deprecated and ignored. If the WWID of a path suddenly changes, multipathd handles it as if it was removed and then added again.

`remove_retries` This sets how many times multipath will retry removing a device that is in-use. Between each attempt, multipath will sleep 1 second.

The default is: 0

`max_sectors_kb` Sets the `max_sectors_kb` device parameter on all path devices and the multipath device to the specified value.

The default is: in `/sys/block/<dev>/queue/max_sectors_kb`

`tors_kb`

`ghost_delay` Sets the number of seconds that multipath will wait after creating a device with only ghost paths before marking it ready for use in systemd. This gives the active paths time to appear before the multipath runs the hardware handler to switch the ghost paths to active ones. Setting this to 0 or no makes multipath immediately mark a device with only ghost paths as ready.

The default is: no

`enable_foreign` Enables or disables foreign libraries (see section FOREIGN MULTIPATH SUPPORT below). The value is a regular expression; foreign libraries are loaded if their name (e.g. "nvme") matches the expression. By default,

no foreign libraries are enabled. Set this to "nvme" to enable NVMe native multipath support, or "\*" to enable all foreign libraries.

The default is: "NONE"

`recheck_wwid` If set to yes, when a failed path is restored, its `wwid` is rechecked. If the `wwid` has changed, the path is removed from the current multipath device, and re-added as a new path. Multipathd will also recheck a path's `wwid` if it is manually re-added. This option only works for SCSI devices that are configured to use the default `uid_attribute`, `ID_SERIAL`, or `sysfs` for getting their `wwid`.

The default is: no

#### blacklist and blacklist\_exceptions sections

The blacklist section is used to exclude specific devices from the multipath topology. It is most commonly used to exclude local disks or non-disk devices (such as LUNs for the storage array controller) from being handled by multipath-tools.

In the blacklist and blacklist\_exceptions sections, starting a quoted value with an exclamation mark "!" will invert the matching of the rest of the regular expression. For instance, "!^sd[a-z]" will match all values that do not start with "sd[a-z]". The exclamation mark can be escaped "\!" to match a literal ! at the start of a regular expression.

Note: The exclamation mark must be inside quotes, otherwise it will be treated as starting a comment.

The blacklist\_exceptions section is used to revert the actions of the blacklist section. This allows one to selectively include ("whitelist") devices which would normally be excluded via the blacklist section. A common usage is to blacklist "everything" using a catch-all regular expression, and create specific blacklist\_exceptions entries for those devices that should be handled by multipath-tools.

The following keywords are recognized in both sections. The defaults are empty unless explicitly stated.

**devnode** Regular expression matching the device nodes to be excluded/included.

The default blacklist consists of the regular expression `"!(sd[a-z]|dasd[a-z]|nvme[0-9])"`. This causes all device types other than scsi, dasd, and nvme to be excluded from multipath handling by default.

**wwid** Regular expression for the World Wide Identifier of a device to be excluded/included.

**device** Subsection for the device description. This subsection recognizes the vendor and product keywords. Both are regular expressions. For a full description of these keywords please see the devices section description.

**property** Regular expression for an udev property. All devices that have matching udev properties will be excluded/included. The handling of the property keyword is special, because if a property blacklist\_exception is set, devices must have at least one whitelisted udev property; otherwise they're treated as blacklisted, and the message "blacklisted, udev property missing" is displayed in the logs. For example, setting the property blacklist\_exception to `(SCSI_IDENT_ID_WWN)`, will cause well-behaved SCSI devices and devices that provide a WWN (World Wide Number) to be included, and all others to be excluded. This works to exclude most non-multipathable devices. Note: The behavior of this option has changed in multipath-tools 0.8.2 compared to previous versions. Blacklisting by missing properties is only applied to devices which do have the property specified by uid\_attribute (e.g. ID\_SERIAL) set. Previously, it was applied to every device, possibly causing devices to be blacklisted because of temporary I/O error conditions.

protocol Regular expression for the protocol of a device to be excluded/included.

The protocol strings that multipath recognizes are scsi:fc, scsi:spi, scsi:ssa, scsi:sbp, scsi:srp, scsi:iscsi, scsi:sas, scsi:adt, scsi:ata, scsi:unspec, nvme:pcie, nvme:rdma, nvme:fc, nvme:tcp, nvme:loop, nvme:apple-nvme, nvme:unspec, ccw, cciss, and undef.

The protocol that a path is using can be viewed by running `multipathd show paths` format "%d %P"

For every device, these 5 blacklist criteria are evaluated in the order "property, devnode, device, protocol, wwid". If a device turns out to be blacklisted by any criterion, it's excluded from handling by `multipathd`, and the later criteria aren't evaluated any more. For each criterion, the whitelist takes precedence over the blacklist if a device matches both.

Note: Besides the blacklist and whitelist, other configuration options such as `find_multipaths` have an impact on whether or not a given device is handled by `multipath-tools`.

## multipaths section

The `multipaths` section allows setting attributes of multipath maps. The attributes that are set via the `multipaths` section (see list below) take precedence over all other configuration settings, including those from the `overrides` section.

The only recognized attribute for the `multipaths` section is the `multi?path` subsection. If there are multiple multipath subsections matching a given WWID, the contents of these sections are merged, and settings from later entries take precedence.

The multipath subsection recognizes the following attributes:

`wwid` (Mandatory) World Wide Identifier. Detected multipath maps are matched against this attribute. Note that, unlike the `wwid` attribute in the blacklist section, this is not a regular expression or a substring; WWIDs must match exactly inside the `multipaths` section.

alias        Symbolic name for the multipath map. This takes precedence over an entry for the same WWID in the bindings\_file.

The following attributes are optional; if not set the default values are taken from the overrides, devices, or defaults section:

- path\_grouping\_policy
- path\_selector
- prio
- prio\_args
- failback
- rr\_weight
- no\_path\_retry
- rr\_min\_io
- rr\_min\_io\_rq
- flush\_on\_last\_del
- features
- reservation\_key
- user\_friendly\_names
- deferred\_remove
- san\_path\_err\_threshold
- san\_path\_err\_forget\_rate
- san\_path\_err\_recovery\_time
- marginal\_path\_err\_sample\_time
- marginal\_path\_err\_rate\_threshold
- marginal\_path\_err\_recheck\_gap\_time
- marginal\_path\_double\_failed\_time
- delay\_watch\_checks
- delay\_wait\_checks
- skip\_kpartx
- max\_sectors\_kb
- ghost\_delay

devices section

multipath-tools have a built-in device table with reasonable defaults

for more than 100 known multipath-capable storage devices. The devices section can be used to override these settings. If there are multiple matches for a given device, the attributes of all matching entries are applied to it. If an attribute is specified in several matching device subsections, later entries take precedence. Thus, entries in files under `config_dir` (in reverse alphabetical order) have the highest precedence, followed by entries in `multipath.conf`; the built-in hardware table has the lowest precedence. Inside a configuration file, later entries have higher precedence than earlier ones.

The only recognized attribute for the devices section is the device subsection. Devices detected in the system are matched against the device entries using the vendor, product, and revision fields, which are all POSIX Extended regular expressions (see `regex(7)`).

The vendor, product, and revision fields that `multipath` or `multipathd` detect for devices in a system depend on the device type. For SCSI devices, they correspond to the respective fields of the SCSI INQUIRY page. In general, the command `'multipathd show paths format "%d %s'` command can be used to see the detected properties for all devices in the system.

The device subsection recognizes the following attributes:

`vendor` (Mandatory) Regular expression to match the vendor name.

`product` (Mandatory) Regular expression to match the product name.

`revision` Regular expression to match the product revision. If not specified, any revision matches.

`product_blacklist`

Products with the given vendor matching this string are blacklisted. This is equivalent to a device entry in the blacklist section with the vendor attribute set to this entry's vendor, and the product attribute set to the value of `product_blacklist`.

`alias_prefix` The `user_friendly_names` prefix to use for this device

type, instead of the default "mpath".

`vpd_vendor` The vendor specific vpd page information, using the vpd page abbreviation. The vpd page abbreviation can be found by running `sg_vpd -e`. multipathd will use this information to gather device specific information that can be displayed with the `%g` wildcard for the multipathd show maps format and multipathd show paths format commands. Currently only the hp3par vpd page is supported.

`hardware_handler` The hardware handler to use for this device type. The following hardware handler are implemented:

- 1 `emc` (Hardware-dependent) Hardware handler for DGC class arrays as CLARiiON CX/AX and EMC VNX and Unity families.
- 1 `rdac` (Hardware-dependent) Hardware handler for LSI/Engenio/NetApp RDAC class as NetApp SANtricity E/EF Series, and OEM arrays from IBM DELL SGI STK and SUN.
- 1 `hp_sw` (Hardware-dependent) Hardware handler for HP/COMPAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode exclusively.
- 1 `alua` (Hardware-dependent) Hardware handler for SCSI-3 ALUA compatible arrays.
- 1 `ana` (Hardware-dependent) Hardware handler for NVMe ANA compatible arrays.

The default is: <unset>

Important Note: Linux kernels 4.3 and newer automatically attach a device handler to known devices (which includes all devices supporting SCSI-3 ALUA) and disallow changing the handler afterwards. Setting `hardware_handler` for such devices on these kernels has no effect.

The following attributes are optional; if not set the default values



are taken from the defaults section:

path\_grouping\_policy

uid\_attribute

getuid\_callout

path\_selector

path\_checker

prio

prio\_args

features

failback

rr\_weight

no\_path\_retry

rr\_min\_io

rr\_min\_io\_rq

fast\_io\_fail\_tmo

dev\_loss\_tmo

eh\_deadline

flush\_on\_last\_del

user\_friendly\_names

retain\_attached\_hw\_handler

detect\_prio

detect\_checker

deferred\_remove

san\_path\_err\_threshold

san\_path\_err\_forget\_rate

san\_path\_err\_recovery\_time

marginal\_path\_err\_sample\_time

marginal\_path\_err\_rate\_threshold

marginal\_path\_err\_recheck\_gap\_time

marginal\_path\_double\_failed\_time

delay\_watch\_checks

delay\_wait\_checks

skip\_kpartx

max\_sectors\_kb

ghost\_delay

all\_tg\_pt

#### overrides section

The overrides section recognizes the following optional attributes; if not set the values are taken from the devices or defaults sections:

path\_grouping\_policy

uid\_attribute

getuid\_callout

path\_selector

path\_checker

alias\_prefix

features

prio

prio\_args

failback

rr\_weight

no\_path\_retry

rr\_min\_io

rr\_min\_io\_rq

flush\_on\_last\_del

fast\_io\_fail\_tmo

dev\_loss\_tmo

eh\_deadline

user\_friendly\_names

retain\_attached\_hw\_handler

detect\_prio

detect\_checker

deferred\_remove

san\_path\_err\_threshold

san\_path\_err\_forget\_rate

san\_path\_err\_recovery\_time

marginal\_path\_err\_sample\_time

marginal\_path\_err\_rate\_threshold  
marginal\_path\_err\_recheck\_gap\_time  
marginal\_path\_double\_failed\_time  
delay\_watch\_checks  
delay\_wait\_checks  
skip\_kpartx  
max\_sectors\_kb  
ghost\_delay  
all\_tg\_pt

The overrides section also recognizes the optional protocol subsection, and can contain multiple protocol subsections. Path devices are matched against the protocol subsection using the mandatory type attribute. Attributes in a matching protocol subsection take precedence over attributes in the rest of the overrides section. If there are multiple matching protocol subsections, later entries take precedence.

protocol subsection

The protocol subsection recognizes the following mandatory attribute:

**type** The protocol string of the path device. The possible values are scsi:fc, scsi:spi, scsi:ssa, scsi:sbp, scsi:srp, scsi:iscsi, scsi:sas, scsi:adt, scsi:ata, scsi:unspec, nvme:pcie, nvme:rdma, nvme:fc, nvme:tcp, nvme:loop, nvme:apple-nvme, nvme:unspec, ccw, cciss, and undef. This is not a regular expression. The path device protocol string must match exactly. The protocol that a path is using can be viewed by running `multipathd show paths for? mat "%d %P"`

The following attributes are optional; if not set, the default values are taken from the overrides, devices, or defaults section:

fast\_io\_fail\_tmo  
dev\_loss\_tmo  
eh\_deadline

## WWID generation

Multipath uses a World Wide Identification (WWID) to determine which paths belong to the same device. Each path presenting the same WWID is assumed to point to the same device.

The WWID is generated by four methods (in the order of preference):

`uid_attrs` The WWID is derived from udev attributes by matching the device node name; cf `uid_attrs` above.

`getuid_callout` Use the specified external program; cf `getuid_callout` above. Care should be taken when using this method; the external program needs to be loaded from disk for execution, which might lead to deadlock situations in an all-paths-down scenario.

`uid_attribute` Use the value of the specified udev attribute; cf `uid_attribute` above. This method is preferred to `getuid_callout` as multipath does not need to call any external programs here. However, under certain circumstances udev might not be able to generate the requested variable.

`sysfs` Try to determine the WWID from sysfs attributes. For SCSI devices, this means reading the Vital Product Data (VPD) page "Device Identification" (0x83).

The default settings (using udev and `uid_attribute` configured from the built-in hardware table) should work fine in most scenarios. Users who want to enable uevent merging must set `uid_attrs`.

## Shaky paths detection

A common problem in SAN setups is the occurrence of intermittent errors: a path is unreachable, then reachable again for a short time, disappears again, and so forth. This happens typically on unstable interconnects. It is undesirable to switch pathgroups unnecessarily on such frequent, unreliable events. multipathd supports three different methods for detecting this situation and dealing with it. All methods share the same basic mode of operation: If a path is found to be "shaky" or "flipping", and appears to be in healthy status, it is not reinstated

(put back to use) immediately. Instead, it is placed in the "delayed" state and watched for some time, and only reinstated if the healthy state appears to be stable. If the `marginal_pathgroups` option is set, the path will be reinstated immediately, but placed in a special pathgroup for marginal paths. Marginal pathgroups will not be used until all other pathgroups have been tried. At the time when the path would normally be reinstated, it will be returned to its normal pathgroup. The logic of determining "shaky" condition, as well as the logic when to reinstate, differs between the three methods.

#### "delay\_checks" failure tracking

This method is deprecated and mapped to the `san_path_err` method. See the `delay_watch_checks` and `delay_wait_checks` options above for more information.

#### "marginal\_path" failure tracking

If a second failure event (good->bad transition) occurs within `marginal_path_double_failed_time` seconds after a failure, high-frequency monitoring is started for the affected path: I/O is sent at a rate of 10 per second. This is done for `marginal_path_err_sample_time` seconds. During this period, the path is not reinstated. If the rate of errors remains below `marginal_path_err_rate_threshold` during the monitoring period, the path is reinstated. Otherwise, it is kept in failed state for `marginal_path_err_recheck_gap_time`, and after that, it is monitored again. For this method, time intervals are measured in seconds.

#### "san\_path\_err" failure tracking

`multipathd` counts path failures for each path. Once the number of failures exceeds the value given by `san_path_err_threshold`, the path is not reinstated for `san_path_err_recovery_time` seconds. While counting failures, `multipathd` "forgets" one past failure every `san_path_err_forget_rate` ticks; thus if errors don't occur more often than once in the forget rate interval, the failure count doesn't increase and the threshold is never

reached. Ticks are the time between path checks by multipathd, which is variable and controlled by the polling\_interval and max\_polling\_interval parameters.

#### "FPIN " failure tracking

Fibre channel fabrics can notify hosts about fabric-level issues such as integrity failures or congestion with so-called Fabric Performance Impact Notifications (FPINs). On receiving the fpin notifications through ELS multipathd will move the affected path and port states to marginal.

This method is deprecated in favor of the "marginal\_path" failure tracking method, and only offered for backward compatibility.

See the documentation of the individual options above for details. It is strongly discouraged to use more than one of these methods for any given multipath map, because the two concurrent methods may interact in unpredictable ways. If the "marginal\_path" method is active, the "san\_path\_err" parameters are implicitly set to 0.

#### FOREIGN MULTIPATH SUPPORT

multipath and multipathd can load "foreign" libraries to add support for other multipathing technologies besides the Linux device mapper. Currently this support is limited to printing detected information about multipath setup. In topology output, the names of foreign maps are prefixed by the foreign library name in square brackets, as in this example:

```
# multipath -ll
uuid.fedcba98-3579-4567-8765-123456789abc [nvme]:nvme4n9 NVMe,Some NVMe controller,FFFFFFFF
size=167772160 features='n/a' hwhandler='ANA' wp=rw
|-+- policy='n/a' prio=50 status=optimized
|`- 4:38:1 nvme4c38n1 0:0 n/a optimized live
`-+- policy='n/a' prio=50 status=optimized
  `- 4:39:1 nvme4c39n1 0:0 n/a optimized live
```

The "nvme" foreign library provides support for NVMe native multipathing in the kernel. It is part of the standard multipath package.

## KNOWN ISSUES

The usage of `queue_if_no_path` option can lead to D state processes being hung and not killable in situations where all the paths to the LUN go offline. It is advisable to use the `no_path_retry` option instead.

The use of `queue_if_no_path` or `no_path_retry` might lead to a deadlock if the `dev_loss_tmo` setting results in a device being removed while I/O is still queued. The multipath daemon will update the `dev_loss_tmo` setting accordingly to avoid this deadlock. Hence if both values are specified the order of precedence is `no_path_retry`, `queue_if_no_path`, `dev_loss_tmo`.

## SEE ALSO

`udev(8)`, `dmsetup(8)`, `multipath(8)`, `multipathd(8)`.

## AUTHORS

`multipath-tools` was developed by Christophe Varoqui, <christophe.varoqui@opensvc.com> and others.

Linux

2021-09-08

MULTIPATH.CONF(5)