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 multipath -t or multi?

   pathd 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 ig?
nored until the end of the line. Inside a quoted string, '#' and '!' are normal charac?
ters, 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", "rhabar?
ber", 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 when?
                ever 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 sec?
                tion WWID generation below. Attributes set in this section take prece?
                dence 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 de?
                vice-specific settings for all devices.

defaults section

The defaults section recognizes the following keywords:

verbosity   Default verbosity. Higher values increase the verbosity level. Valid lev?

        els 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 * 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 de?

        vice, 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. Di?

        rectory 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 ker?

        nel 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 con?

            trolled 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.

io-affinity 0

> (Since 5.11 kernel) Choose the path for the next bunch of I/O
> based on a CPU to path mapping the user passes in and what
> CPU we are executing on.

The default is: service-time 0

path_grouping_policy

> The default path grouping policy to apply to unspecified multipaths. Pos?
> sible 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 deter?
> > mined by callout programs specified as a global, per-con?
> > troller 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 effiency. Moreover, it's an alternative method to con?
figure 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 de?

vice, 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_at‐ tribute is set to the empty string, WWID determination is done using the sysfs method rather then 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 re‐ turn 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 op‐ tional 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/Enge‐ nio/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/COM?

   PAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode ex?

   clusively.

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. Re?

   quires 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 Dat?

   aCore 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 al?

gorithm is const.

prio_args   Arguments to pass to to the prio function. This only applies to certain

   prioritizers:

weighted   Needs a value of the form "<hbtl|devname|serial|wwn> <regex1>

   <prio1> <regex2> <prio2> ..."

   hbtl   Regex can be of SCSI H:B:T:L format. For example:

      1:0:.:. , *:0:0:.

   devname Regex can be of device name format. For example: sda

      , sd.e

   serial Regex can be of serial number format. For example:

      .*J1FR.*324 . The serial can be looked up through

sysfs or by running multipathd 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.redhat.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 con‐
tinuously, used to calculate the average path la‐
tency. Valid Values: Integer, [2, 200].

base_num

The base number value of logarithmic scale, used to
partition different priority ranks. Valid Values: In‐
teger, [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 INQUIRY, in ms.

iet

preferredip=...

(Mandatory) Th preferred IP address, in dotted deci‐

mal 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 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) re? quest-based, respectively. The default depends on the kernel parameter dm_mod.use_blk_mq. It is mq if the latter is set, and rq otherwise.

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 re? spond. The asynchronous checkers (tur and directio) will not pause multi? pathd. Instead, multipathd will check for a response once per second, un? til 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 exclu?

>   sively.

rdac   (Hardware-dependent) Check the path state for LSI/Engenio/Ne?

>   tApp RDAC class as NetApp SANtricity E/EF Series, and OEM ar?

>   rays 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 pri?

oritizer.

immediate   Immediately failback 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 Ac?

>   tive/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 mul? tipath 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 maxi? mum number of paths plus 32, if that number is greated 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 re? stored, 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 prioritizers, in sec‐

onds.   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 be 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 automatically adjusted to the

overall retry interval no_path_retry * polling_interval if a number of

retries 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 er?

ror 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 persistent 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 re?

stored. 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 appli?

cation client to the device server to identify the I_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 reg‐

istration 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 mpathpersist on these arrays. Set‐

ting this option is identical to calling mpathpersist with --param-allt‐

gpt

The default is: no

retain_attached_hw_handler

(Obsolete for kernels >= 4.3) If set to yes and the SCSI layer has al‐

ready 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 "re‐

tain_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 priori‐

tizer 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 sig‐ nificant CPU pressure.

The default is: no

strict_timing   If set to yes , multipathd will start a new path checker loop after ex‐ actly 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 de‐ vices 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 be‐ tween device name and partition number to construct the partition device name.   Otherwise (i.e. if this parameter is unset or has the value "/UN‐ SET/"), the behavior depends on the map name: if it ends in a digit, a "p" is inserted between name and partition number; otherwise, the parti‐ tion 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 cur‐ rent 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 al?
              phabetically for file ending in ".conf" and it will read configuration
              information from them, just as if it was in /etc/multipath.conf.   con?
              fig_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 then the san_path_err_threshold,
              then the path will not reinstate 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 reinstantate. 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 reinstante 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 reinstante 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 con?
tinuous 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_dou?
ble_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 mar?
ginal_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 marginal_path_err_recheck_gap_time seconds unless there
is only one active path. After marginal_path_err_recheck_gap_time ex?
pires, the path will be requeueed for rechecking. If checking result is
good enough, the path will be reinstated. See "Shaky paths detection" be?
low.

The default is: no

marginal_path_err_rate_threshold

The error rate threshold as a permillage (1/1000). One of the four param?
eters of supporting path check based on accounting IO error such as in?
termittent 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_sam?

ple_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_thresh?

old 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 requeueed 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 de?

lay_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   de?

lay_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_recov?

ery_time and san_path_err_threshold options, and "Shaky paths detection"

below for more information.

The default is: no

marginal_pathgroups

If set to no, the delay_*_checks, marginal_path_*, and san_path_err_* op?

tions 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 yes, when one of the marginal path detecting methods determines that a path is marginal, it will be reinstated and placed in a seperate pathgroup that will only be used after all the non-marginal pathgroups have been tried first. This prevents the possibility of IO errors occur? ing while marginal paths are still usable. After the path has been moni? tored for the configured time, and is declared healthy, it will be re? turned to its normal pathgroup. See "Shaky paths detection" below for more information.

The default is: no

find_multiraths This option controls whether multipath and multipathd try to create mul? tipath 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 possi? ble:

strict   Both multipath and multipathd treat only such devices as multi? path devices which have been part of a multipath map previ? ously, and which are therefore listed in the wwids_file. Users can manually set up multipath maps using the multipathd add map command. 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 de? vice 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 de? vice 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 encounted 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.

The default is: strict

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 recom?

mended 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 de?

vice is created, to receive a change event from udev for the device, be?

fore 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 may 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 mul?

tipath device to the specified value.

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

ghost_delay   Sets the number of seconds that multipath will wait after creating a de?

vice 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 SUP?

PORT 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 de?

vice, 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 excla?

mation mark "!" will invert the matching of the rest of the regular expression. For in?

stance, "!^sd[a-z]" will match all values that do not start with "sd[a-z]". The exclama?

tion 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 ex?

cluded 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 ex?

plicitly 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 ex?

               cluded/included.

device   Subsection for the device description. This subsection recognizes the

               vendor and product keywords. Both are regular expressions. For a full de?

               scription 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 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.

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_at? tribute (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.

The default blacklist exception is: (SCSI_IDENT_|ID_WWN), causing well-behaved SCSI devices and devices that provide a WWN (World Wide Number) to be included, and all others to be excluded.

protocol   Regular expression for the protocol of a device to be excluded/included. The protocol strings that multipath recognizes are scsi:fcp, scsi:spi, scsi:ssa, scsi:sbp, scsi:srp, scsi:iscsi, scsi:sas, scsi:adt, scsi:ata, scsi:unspec, ccw, cciss, nvme, 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 the order "property, devnode, device, protocol, wwid". If a device turns out to be blacklisted by any crite? rion, 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 de? vice matches both.

Note: Besides the blacklist and whitelist, other configuration options such as find_multi? paths 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 config? uration settings, including those from the overrides section.

The only recognized attribute for the multipaths section is the multipath subsection. If there are multiple multipath subsections matching a given WWID, the contents of these sec? tions 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 agains 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 a an en‐

           try 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 sub‐sections, later entries take precedence. Thus, entries in files under config_dir (in re‐verse alphabetical order) have the highest precedence, followed by entries in multi‐path.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 abbrevia‐
                 tion. The vpd page abbreviation can be found by running sg_vpd -e. mul‐
                 tipathd will use this information to gather device specific information
                 that can be displayed with the %g wilcard 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 compat?

 ible 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_han?

dler 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

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

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

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 occurence of intermittent errors: a path is unreach‐
able, 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 de‐
tecting this situation and dealing with it. All methods share the same basic mode of oper‐
ation: If a path is found to be "shaky" or "flipping", and appears to be in healthy sta‐
tus, it is not reinstated (put back to use) immediately. Instead, it is placed in the "de‐
layed" 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 reinstated immedi‐
ately, 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 nor‐
mally 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 meth‐
ods.

"delay_checks" failure tracking

        This method is deprecated and mapped to the "san_path_err" method.   See the de‐

        lay_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_dou‐

        ble_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 mar?

ginal_path_err_sample_time seconds. During this period, the path is not rein?

stated. If the rate of errors remains below marginal_path_err_rate_threshold dur?

ing 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 oc?

cur more often then 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.

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 dis?

couraged 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 multi?

pathing 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 exam?

ple:

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 kil?

lable 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)