Red Hat Enterprise Linux 9

Configuring device mapper multipath

Using the Device Mapper Multipath feature

Last Updated: 2022-06-02

Red Hat Enterprise Linux 9 Configuring device mapper multipath

Using the Device Mapper Multipath feature

Legal Notice

Abstract

This documentation collection provides instructions on how to configure and manage the Device Mapper Multipath (DM-Multipath) feature on Red Hat Enterprise Linux 9.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better.

- For simple comments on specific passages:
    1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
    2. Use your mouse cursor to highlight the part of text that you want to comment on.
    3. Click the **Add Feedback** pop-up that appears below the highlighted text.
    4. Follow the displayed instructions.
- For submitting feedback via Bugzilla, create a new ticket:
    1. Go to the Bugzilla website.
    2. As the Component, use **Documentation**.
    3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
    4. Click **Submit Bug**.

CHAPTER 1. OVERVIEW OF DEVICE MAPPER MULTIPATHING

With Device mapper multipathing (DM Multipath), you can configure multiple I/O paths between server nodes and storage arrays into a single device. These I/O paths are physical Storage Area Network (SAN) connections that can include separate cables, switches, and controllers. Multipathing aggregates the I/O paths and creates a new device that consists of the aggregated paths.

DM Multipath provides:

Redundancy

DM Multipath can provide failover in an active/passive configuration. In an active/passive configuration, only half the paths are used at any time for I/O. If any element of an I/O path such as the cable, switch, or controller fails, DM Multipath switches to an alternate path.

Improved Performance

DM Multipath can be configured in an active/active mode, where I/O is spread over the paths in a round-robin fashion. In some configurations, DM Multipath can detect loading on the I/O paths and dynamically rebalance the load.

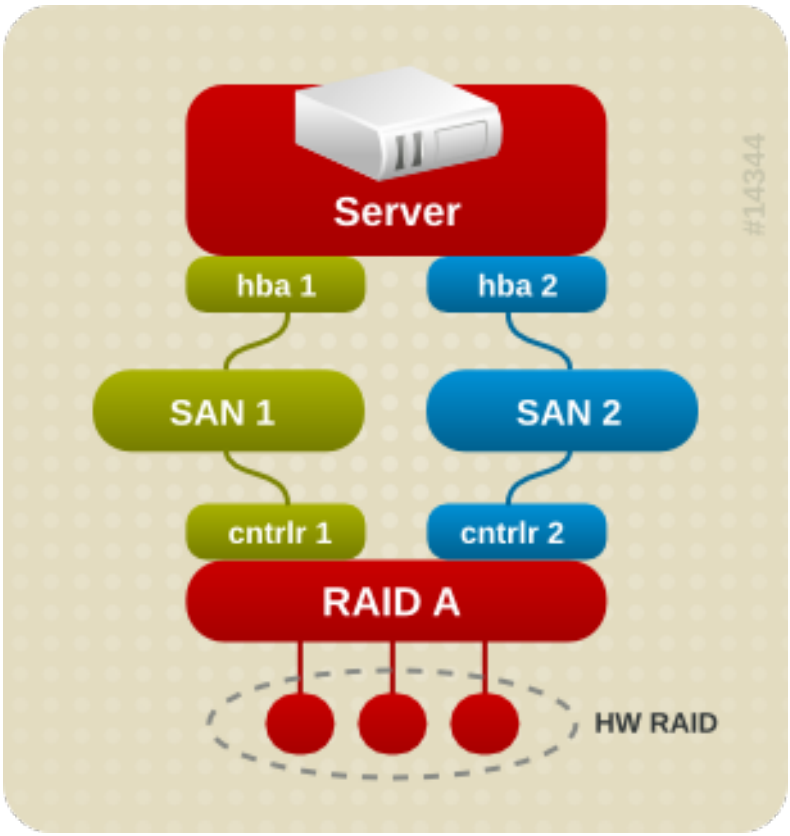
1.1. ACTIVE/PASSIVE MULTIPATH CONFIGURATION WITH ONE RAID DEVICE

In this configuration, there are two Host Bus Adapters (HBAs) on the server, two SAN switches, and two RAID controllers. Following are the possible failure in this configuration:

- HBA failure
- Fibre Channel cable failure
- SAN switch failure
- Array controller port failure

With DM Multipath configured, a failure at any of these points causes DM Multipath to switch to the alternate I/O path. The following image describes the configuration with two I/O paths from the server to a RAID device. Here, there is one I/O path that goes through **hba1**, **SAN1**, and **cntrlr1** and a second I/O path that goes through **hba2**, **SAN2**, and **cntrlr2**.

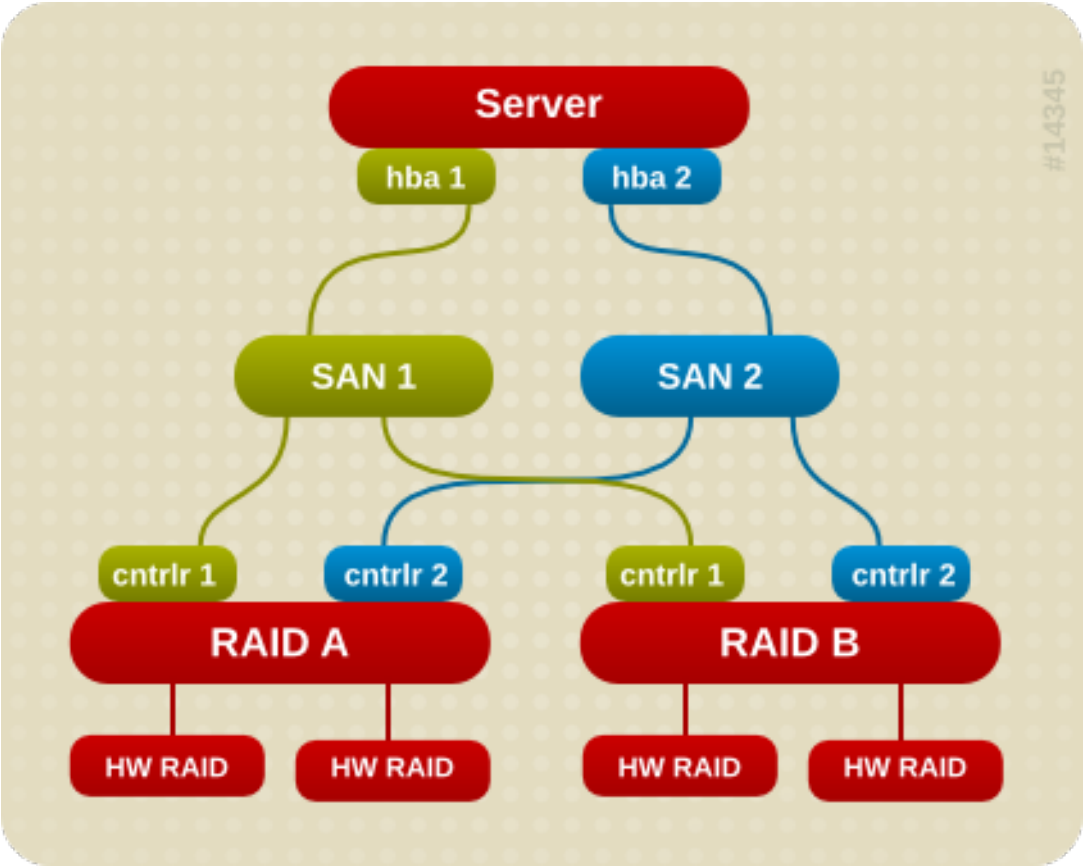
Figure 1.1. Active/Passive multipath configuration with one RAID device



1.2. ACTIVE/PASSIVE MULTIPATH CONFIGURATION WITH TWO RAID DEVICES

In this configuration, there are two HBAs on the server, two SAN switches, and two RAID devices with two RAID controllers each. With DM Multipath configured, a failure at any of the points of the I/O path to either of the RAID devices causes DM Multipath to switch to the alternate I/O path for that device. The following image describes the configuration with two I/O paths to each RAID device. Here, there are two I/O paths to each RAID device.

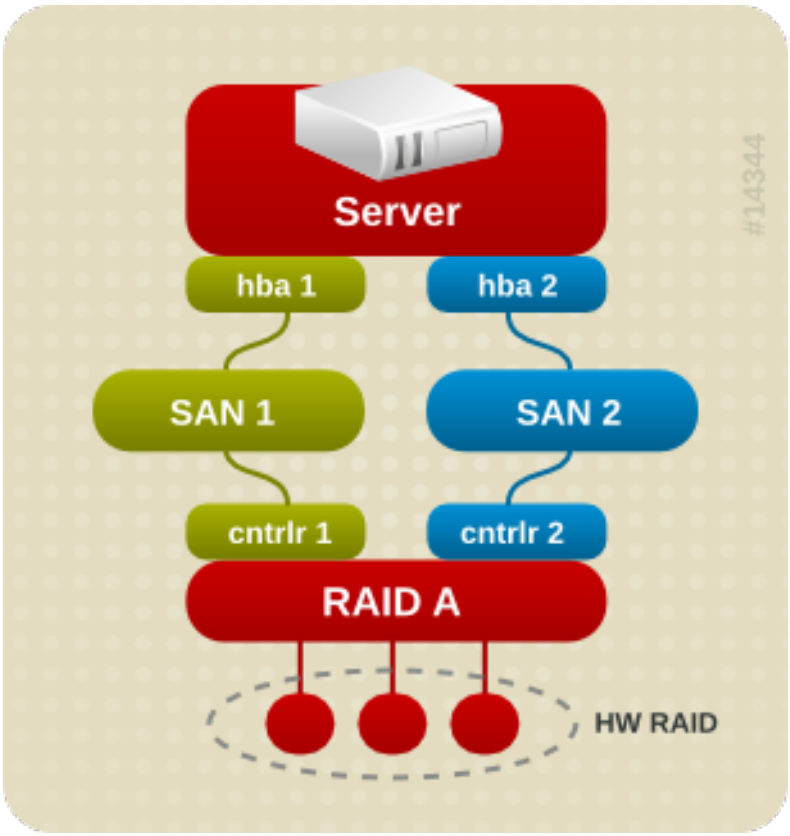
Figure 1.2. Active/Passive multipath configuration with two RAID device



1.3. ACTIVE/ACTIVE MULTIPATH CONFIGURATION WITH ONE RAID DEVICE

In this configuration, there are two HBAs on the server, two SAN switches, and two RAID controllers. The following image describes the configuration with two I/O paths from the server to a storage device. Here, I/O can be spread among these two paths.

Figure 1.3. Active/Active multipath configuration with one RAID device



1.4. DM MULTIPATH COMPONENTS

The following table describes the DM Multipath components.

Table 1.1. Components of DM Multipath

Component	Description
dm_multipath kernel module	Reroutes I/O and supports failover for paths and path groups.
mpathconf utility	Configures and enables device mapper multipathing.
multipath command	Lists and configures the multipath devices. It is also executed by **udev** whenever a block device is added, to determine if the device should be part of a multipath device or not.
multipathd daemon	Automatically creates and removes multipath devices and monitors paths; as paths fail and come back, it may update the multipath device. Allows interactive changes to multipath devices. Reload the service if there are any changes to the **/etc/multipath.conf** file.

kpartx command	Creates device mapper devices for the partitions on a device. This command is automatically executed by **udev** when multipath devices are created to create partition devices on top of them. The **kpartx** command is provided in its own package, but the **device-mapper-multipath** package depends on it.
mpathpersist	Sets up **SCSI-3** persistent reservations on multipath devices. This command works similarly to the way **sg_persist** works for SCSI devices that are not multipathed, but it handles setting persistent reservations on all paths of a multipath device. It coordinates with **multipathd** to ensure that the reservations are set up correctly on paths that are added later. To use this functionality, the **reservation_key** attribute must be defined in the **/etc/multipath.conf** file. Otherwise the **multipathd** daemon will not check for persistent reservations for newly discovered paths or reinstated paths.

1.5. THE MULTIPATH COMMAND

The **multipath** command is used to detect and combine multiple paths to devices. It provides a variety of options you can use to administer your multipathed devices.

The following table describes some options of the **multipath** command that you may find useful.

Table 1.2. Useful multipath command options

Option	Description
-l	Display the current multipath configuration gathered from **sysfs** and the device mapper.
-ll	Display the current multipath configuration gathered from **sysfs**, the device mapper, and all other available components on the system.
-f *device*	Remove the named multipath device.
-F	Remove all unused multipath devices.
-w *device*	Remove the **wwid** of the specified device from the **wwids** file.
-W	Reset the **wwids** file to include only the current multipath devices.

1.6. MULTIPATH COMMAND OUTPUT

When you create, modify, or list a multipath device, you get a display of the current device setup. The format is as follows.

- For each multipath device:

```
action_if_any: alias (wwid_if_different_from_alias) dm_device_name_if_known
vendor,product size=size features='features' hwhandler='hardware_handler'
wp=write_permission_if_known
```

- For each path group:

```
-+- policy='scheduling_policy' prio=prio_if_known status=path_group_status_if_known
```

- For each path:

```
 `- host:channel:id:lun devnode major:minor dm_status_if_known path_status online_status
```

For example, the output of a multipath command might appear as follows:

```
3600d0230000000000e13955cc3757800 dm-1 WINSYS,SF2372
size=269G features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=active
| `- 6:0:0:0 sdb 8:16  active ready  running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 7:0:0:0 sdf 8:80  active ready  running
```

If the path is up and ready for I/O, the status of the path is **ready** or **ghost**. If the path is down, the status is **faulty** or **shaky**. The path status is updated periodically by the **multipathd** daemon based on the polling interval defined in the **/etc/multipath.conf** file.

Additional possible path status values are as follows.

- **i/o pending**: The checker is actively checking this path, and the state will be updated shortly.
- **i/o timeout**: This is the same as **faulty**. It lets the user know that the checker did not return either success or failure before the timeout period.
- **removed**: The path has been removed from the system, and will shortly be removed from the multipath device. It is treated the same as **faulty**.
- **wild**: **multipathd** was unable to run the path checker, because of an internal error or configuration issue. This is roughly the same as **faulty**, except multipath will skip many actions on the path.
- **unchecked**: The path checker has not run on this path, either because it has just been discovered, it does not have an assigned path checker, or the path checker encountered an error. This is treated the same as **wild**.
- **delayed**: The path checker returns that the path is up, but multipath is delaying the reinstatement of the path because the path has recently failed multiple times and multipath has been configured to delay paths in this case.

In terms of the kernel, the dm status is similar to the path status The **active** dm state covers the **ready** and **ghost** path states. The **pending** path state has no equivalent dm state. All other path states map to the **failed** dm state. The dm state will retain its current status until the path checker has completed.

The possible values for **online_status** are **running** and **offline**. The **offline** status means that this SCSI device has been disabled.

NOTE

When you create or modify a multipath device, multipath prints the device configuration. However, some of the features, for example, write permissions, and other feature information might be unknown. There might be a difference between the output and the features that you selected during creation or modification. This is normal behaviour. List the device after creation to view the correct state.

1.7. DISPLAYING MULTIPATH CONFIGURATION

You can use the **-l** and **multipath** command to display the current multipath configuration. The **-l** option displays multipath topology gathered from information in **sysfs** and the device mapper. The **-ll** option displays the information the **-l** option displays in addition to all other available components of the system.

When displaying the multipath configuration, you can specify a verbosity level with the **-v** option of the **multipath** command. Specifying **-v0** yields no output. Specifying **-v1** outputs the created or updated multipath names only, which you can then feed to other tools such as **kpartx**. Specifying **-v2** prints all detected paths, multipaths, and device maps. For even more detailed information, you can also specify **-v3, -v4**, or **-v5**.

The following example shows the output of a **multipath -l** command.

```
# multipath -l
3600d0230000000000e13955cc3757800 dm-1 WINSYS,SF2372
size=269G features='0' hwhandler='0' wp=rw
|-- policy='round-robin 0' prio=1 status=active
| `- 6:0:0:0 sdb 8:16  active ready  running
`-- policy='round-robin 0' prio=1 status=enabled
  `- 7:0:0:0 sdf 8:80  active ready  running
```

The following example shows the output of a **multipath -ll** command.

```
# multipath -ll
3600d0230000000000e13955cc3757801 dm-10 WINSYS,SF2372
size=269G features='0' hwhandler='0' wp=rw
|-- policy='round-robin 0' prio=1 status=enabled
| `- 19:0:0:1 sdc 8:32  active ready  running
`-- policy='round-robin 0' prio=1 status=enabled
  `- 18:0:0:1 sdh 8:112 active ready  running
3600d0230000000000e13955cc3757803 dm-2 WINSYS,SF2372
size=125G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 19:0:0:3 sde 8:64  active ready  running
  `- 18:0:0:3 sdj 8:144 active ready  running
```

1.8. ADDITIONAL RESOURCES

- **multipath(8)** and **multipathd(8)** man pages
- **/etc/multipath.conf** file

CHAPTER 2. MULTIPATH DEVICES

DM Multipath provides a way of organizing the I/O paths logically, by creating a single multipath device on top of the underlying devices. Without DM Multipath, system treats each path from a server node to a storage controller as a separate device, even when the I/O path connects the same server node to the same storage controller.

2.1. MULTIPATH DEVICE IDENTIFIERS

When new devices are under the control of DM Multipath, these devices are created in the **/dev/mapper/** and **/dev/** directory.

NOTE

Any devices of the form **/dev/dm-X** are for internal use only and should never be used by the administrator directly.

The following describes multipath device names:

- When the **user_friendly_names** configuration option is set to **no**, the name of the multipath device is set to World Wide Identifier (WWID). By default, the name of a multipath device is set to its WWID. The device name would be **/dev/mapper/WWID**. It is also created in the **/dev/** directory, named as **/dev/dm-X**.
- Alternately, you can set the **user_friendly_names** option to **yes** in the **/etc/multipath.conf** file. This sets the **alias** in the **multipath** section to a node-unique name of the form **mpathN**. The device name would be **/dev/mapper/mpathN** and **/dev/dm-X**. But the device name is not guaranteed to be the same on all nodes using the multipath device. Similarly, if you set the **alias** option in the **/etc/multipath.conf** file, the name is not automatically consistent across all nodes in the cluster.

NOTE

This should not cause any difficulties if you use LVM to create logical devices from the multipath device. To keep your multipath device names consistent in every node, Red Hat recommends disabling the **user_friendly_names** option.

For example, a node with two HBAs attached to a storage controller with two ports by means of a single unzoned FC switch sees four devices: **/dev/sda**, **/dev/sdb**, **/dev/sdc**, and **/dev/sdd**. DM Multipath creates a single device with a unique WWID that reroutes I/O to those four underlying devices according to the multipath configuration.

In addition to the **user_friendly_names** and **alias** options, a multipath device also has other attributes. You can modify these attributes for a specific multipath device by creating an entry for that device in the **multipaths** section of the **/etc/multipath.conf** file.

Additional resources

- **multipath(8)** and **multipath.conf(8)** man pages
- **/etc/multipath.conf** file
- DM Multipath components

2.2. MULTIPATH DEVICES IN LOGICAL VOLUMES

After creating multipath devices, you can use the multipath device names as you would use a physical device name when creating an Logical volume manager (LVM) physical volume. For example, if **/dev/mapper/mpatha** is the name of a multipath device, the **pvcreate /dev/mapper/mpatha** command marks **/dev/mapper/mpatha** as a physical volume.

You can use the resulting LVM physical device when you create an LVM volume group just as you would use any other LVM physical device.

To filter all the **sd** devices in the **/etc/lvm/lvm.conf** file, add the **filter = ["r/block/", "r/disk/", "r/sd./", "a/./"]** filter in the **devices** section of the file.

NOTE

If you attempt to create an LVM physical volume on a whole device on which you have configured partitions, the **pvcreate** command fails. The Anaconda and Kickstart installation programs create empty partition tables if you do not specify otherwise for every block device. If you want to use the whole device instead of creating a partition, remove the existing partitions from the device. You can remove existing partitions with the **kpartx -d** device command and the **fdisk** utility. If your system has block devices that are greater than 2Tb, use the **parted** utility to remove partitions.

When you create an LVM logical volume that uses **active/passive** multipath arrays as the underlying physical devices, you can optionally include filters in the **/etc/lvm/lvm.conf** file to exclude the disks that underline the multipath devices. This is because if the array automatically changes the active path to the passive path when it receives I/O, multipath will failover and failback whenever LVM scans the passive path, if these devices are not filtered.

The kernel changes the active/passive state by automatically detecting the correct hardware handler to use. For active/passive paths that require intervention to change their state, multipath automatically uses this hardware handler to do so as necessary. If the kernel does not automatically detect the correct hardware handler to use, you can configure which hardware handler to use in the multipath.conf file with the "hardware_handler" option. For **active/passive** arrays that require a command to make the passive path active, LVM prints a warning message when this occurs.

Depending on your configuration, LVM may print any of the following messages:

- LUN not ready:

```
end_request: I/O error, dev sdc, sector 0
sd 0:0:0:3: Device not ready: <6>: Current: sense key: Not Ready
    Add. Sense: Logical unit not ready, manual intervention required
```

- Read failed:

```
/dev/sde: read failed after 0 of 4096 at 0: Input/output error
```

The following are the reasons for the mentioned errors:

- Multipath is not set up on storage devices that are providing active/passive paths to a machine.
- Paths are accessed directly, instead of through the multipath device.

Additional resources

- **lvm.conf** man page
- DM Multipath components

CHAPTER 3. CONFIGURING DM MULTIPATH

You can set up DM Multipath with the **mpathconf** utility. This utility creates or edits the **/etc/multipath.conf** multipath configuration file based on the following scenarios:

- If the **/etc/multipath.conf** file already exists, the **mpathconf** utility will edit it.
- If the **/etc/multipath.conf** file does not exist, the **mpathconf** utility will create the **/etc/multipath.conf** file from scratch.

3.1. CHECKING FOR THE DEVICE-MAPPER-MULTIPATH PACKAGE

Before setting up DM Multipath on your system, ensure that your system is up-to-date and includes the **device-mapper-multipath** package.

Procedure

1. Check if your system includes the the **device-mapper-multipath** package:

```
# rpm -q device-mapper-multipath
device-mapper-multipath-current-package-version
```

   If your system does not include the package, it prints the following:

```
package device-mapper-multipath is not installed
```

2. If your system does not include the package, install it by running the following command:

```
# {PackageManager} install device-mapper-multipath
```

3.2. SETTING UP DM MULTIPATH FOR A BASIC FAILOVER CONFIGURATION

Use the following procedure to set up DM Multipath for a basic failover configuration if you need to edit the **/etc/multipath.conf** file before starting the multipathd daemon.

Procedure

1. Enable the multipath configuration file:

```
# mpathconf --enable
```

2. Edit the **/etc/multipath.conf** file if necessary. The default settings for DM Multipath are compiled into the system and do not need to be explicitly set in the **/etc/multipath.conf** file. The default value of **path_grouping_policy** is set to **failover**, so in this example you do not need to edit the **/etc/multipath.conf** file.

   The initial defaults section of the configuration file configures your system so that the names of the multipath devices are of the form **/dev/mapper/mpathn;** without this setting, the names of the multipath devices would be aliased to the WWID of the device. If you do not want to use user friendly names, you can enter the following command:

```
# mpathconf --enable --user_friendly_names n
```

If you need to edit the multipath configuration file after you have started the multipath daemon, you must execute the **systemctl reload multipathd.service** command for the changes to take effect.

3. Save the configuration file and exit the editor, if necessary.

4. Start the multipath daemon and create the multipath devices:

```
# systemctl start multipathd.service
```

NOTE

If you remove the **device-mapper-multipath** package, this does not remove the **/etc/multipath.conf** file, or any files in the **/etc/multipath** directory, since that directory can contain more that just the files listed currently. You might need to remove those files manually on subsequent installations of the **device-mapper-multipath** package.

3.3. IGNORING LOCAL DISKS WHEN GENERATING MULTIPATH DEVICES

Some machines have local SCSI cards for their internal disks and DM Multipath is not recommended for these devices. If you set the **find_multipaths** configuration parameter to **on**, you do not have to disable multipathing on these devices.

If you do not set the **find_multipaths** configuration parameter to **on**, you can use the following procedure to modify the DM Multipath configuration file to ignore the local disks when configuring multipath.

Procedure

1. Determine which disks are the internal disks. In these examples, **/dev/sda** is the internal disk:

   - Display existing multipath devices:

```
# multipath -v2 -l

SIBM-ESXSST336732LC____F3ET0EP0Q000072428BX1 dm-2 WINSYS,SF2372
size=33 GB features="0" hwhandler="0" wp=rw
`-+- policy='round-robin 0' prio=0 status=active
  |- 0:0:0:0 sda 8:0 active undef running
```

   - Display additional multipath devices that DM Multipath could create:

```
# multipath -v2 -d

: SIBM-ESXSST336732LC____F3ET0EP0Q000072428BX1 undef WINSYS,SF2372
size=33 GB features="0" hwhandler="0" wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 0:0:0:0 sda 8:0  undef ready running
```

2. Edit the **blacklist** section of the **/etc/multipath.conf** file to include this device.

Identify the device using its WWID attribute. Although you could identify the **sda** device using a **devnode** type, that would not be a safe procedure because /**dev**/**sda** is not guaranteed to be the same on reboot.

In the previous example, the WWID of the /**dev**/**sda** device is **SIBM-ESXSST336732LC____F3ET0EP0Q000072428BX1**. To ignore this device, include the following in the /**etc**/**multipath.conf** file:

```
blacklist {
    wwid SIBM-ESXSST336732LC____F3ET0EP0Q000072428BX1
}
```

3. Validate the /**etc**/**multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

4. Reload the /**etc**/**multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

Additional resources

- **multipath.conf(5)** man page

3.4. CONFIGURING ADDITIONAL STORAGE DEVICES

By default, DM Multipath includes support for the most common storage arrays, which support DM Multipath.

Procedure

- View the default configuration value, including supported devices:

  ```
  # multipathd show config
  # multipath -t
  ```

- Optional: To add an additional storage device that is not supported by default as a known multipath device, edit the /**etc**/**multipath.conf** file and insert the appropriate device information.
  The following example ilustrates how to add information about the HP Open-V series. This sets the device to queue for a minute or 12 retries and 5 seconds per retry after all paths have failed.

  ```
  devices {
        device {
              vendor "HP"
  ```

```
        product "OPEN-V"
        no_path_retry 12
    }
}
```

3.5. SETTING UP MULTIPATHING IN THE INITRAMFS FILE SYSTEM

You can set up multipathing in the **initramfs** file system. You do not need to set it up if you are not using the devices that you want multipathed, until after boot leaves the **initramfs** file system.

Prerequisites

- You have configured DM multipath in your system.

Procedure

- Rebuild the **initramfs** file system with the multipath configuration files by running the following command:

```
# dracut --force --add multipath
```

  If you run multipath from the **initramfs** file system and you make any changes to the multipath configuration files, you must rebuild the **initramfs** file system for the changes to take effect. When a root device uses multipath, running the **dracut** command automatically adds the multipath module to the **initramfs**.

- Optional: If you need multipath running in the initramfs, but you are not setting up a multipathed root device, run:

```
# echo add_dracutmodules+=\"multipath\" > /etc/dracut.conf.d/multipath.conf
# dracut --force
```

NOTE

The **dracut** command includes multipath in the **initramfs**, even after multipath is no longer required. To stop including multipath, run:

```
# rm /etc/dracut.conf.d/multipath.conf
# dracut --force
```

CHAPTER 4. ENABLING MULTIPATHING ON NVME DEVICES

You can multipath NVMe devices that are connected to your system over a fabric transport, such as Fibre Channel (FC). You can select between multiple multipathing solutions.

4.1. NATIVE NVME MULTIPATHING AND DM MULTIPATH

NVMe devices support a native multipathing functionality. When configuring multipathing on NVMe, you can select between the standard DM Multipath framework and the native NVMe multipathing.

Both DM Multipath and native NVMe multipathing support the Asymmetric Namespace Access (ANA) multipathing scheme of NVMe devices. ANA identifies optimized paths between the target and the initiator and improves performance.

When native NVMe multipathing is enabled, it applies globally to all NVMe devices. It can provide higher performance, but does not contain all of the functionality that DM Multipath provides. For example, native NVMe multipathing supports only the **failover** and **round-robin** path selection methods.

By default, NVMe multipathing is enabled in Red Hat Enterprise Linux 9 and is the recommended multipathing solution.

4.2. ENABLING NATIVE NVME MULTIPATHING

This procedure enables multipathing on connected NVMe devices using the native NVMe multipathing solution.

Prerequisites

- The NVMe devices are connected to your system.
  For more information on connecting NVMe over fabric transports, see Overview of NVMe over fabric devices.

Procedure

1. Check if native NVMe multipathing is enabled in the kernel:

   ```
   # cat /sys/module/nvme_core/parameters/multipath
   ```

   The command displays one of the following:

   N
   Native NVMe multipathing is disabled.

   Y
   Native NVMe multipathing is enabled.

2. If native NVMe multipathing is disabled, enable it using one of the following methods:

   - Using a kernel option:

     i. Add the **nvme_core.multipath=Y** option on the kernel command line:

     ```
     # grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
     ```

ii. On the 64-bit IBM Z architecture, update the boot menu:

```
# zipl
```

iii. Reboot the system.

- Using a kernel module configuration file:

i. Create the **/etc/modprobe.d/nvme_core.conf** configuration file with the following content:

```
options nvme_core multipath=Y
```

ii. Back up the **initramfs** file system:

```
# cp /boot/initramfs-$(uname -r).img \
    /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

iii. Rebuild the **initramfs** file system:

```
# dracut --force --verbose
```

iv. Reboot the system.

3. Optional: On the running system, change the I/O policy on NVMe devices to distribute the I/O on all available paths:

```
# echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

4. Optional: Set the I/O policy persistently using **udev** rules. Create the **/etc/udev/rules.d/71-nvme-io-policy.rules** file with the following content:

```
ACTION=="add|change", SUBSYSTEM=="nvme-subsystem", ATTR{iopolicy}="round-robin"
```

Verification

1. Check that your system recognizes the NVMe devices:

```
# nvme list

Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------
- ---------------- --------
/dev/nvme0n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme0n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
```

2. List all connected NVMe subsystems:

```
# nvme list-subsys
```

```
nvme-subsys0 - NQN=testnqn
\
 +- nvme0 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme1 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
 +- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

Check the active transport type. For example, **nvme0 fc** indicates that the device is connected over the Fibre Channel transport, and **nvme tcp** indicates that the device is connected over TCP.

3. If you edited the kernel options, check that native NVMe multipathing is enabled on the kernel command line:

```
# cat /proc/cmdline

BOOT_IMAGE=[...] nvme_core.multipath=Y
```

4. Check that DM Multipath reports the NVMe namespaces as, for example, **nvme0c0n1** through **nvme0c3n1**, and *not* as, for example, **nvme0n1** through **nvme3n1**:

```
# multipath -e -ll | grep -i nvme

uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03 [nvme]:nvme0n1 NVMe,Linux,4.18.0-2
| `- 0:0:1    nvme0c0n1 0:0    n/a   optimized live
| `- 0:1:1    nvme0c1n1 0:0    n/a   optimized live
| `- 0:2:1    nvme0c2n1 0:0    n/a   optimized live
  `- 0:3:1    nvme0c3n1 0:0    n/a   optimized live

uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22 [nvme]:nvme0n2 NVMe,Linux,4.18.0-2
| `- 0:0:1    nvme0c0n1 0:0    n/a   optimized live
| `- 0:1:1    nvme0c1n1 0:0    n/a   optimized live
| `- 0:2:1    nvme0c2n1 0:0    n/a   optimized live
  `- 0:3:1    nvme0c3n1 0:0    n/a   optimized live
```

5. If you changed the I/O policy, check that **round-robin** is the active I/O policy on NVMe devices:

```
# cat /sys/class/nvme-subsystem/nvme-subsys0/iopolicy

round-robin
```

Additional resources

- Configuring kernel command-line parameters

4.3. ENABLING DM MULTIPATH ON NVME DEVICES

This procedure enables multipathing on connected NVMe devices using the DM Multipath solution.

Prerequisites

- The NVMe devices are connected to your system.
  For more information on connecting NVMe over fabric transports, see Overview of NVMe over fabric devices.

Procedure

1. Check that native NVMe multipathing is disabled:

```
# cat /sys/module/nvme_core/parameters/multipath
```

   The command displays one of the following:

   N
   Native NVMe multipathing is disabled.

   Y
   Native NVMe multipathing is enabled.

2. If native NVMe multipathing is enabled, disable it:

   a. Remove the **nvme_core.multipath=Y** option from the kernel command line:

```
# grubby --update-kernel=ALL --remove-args="nvme_core.multipath=Y"
```

   b. On the 64-bit IBM Z architecture, update the boot menu:

```
# zipl
```

   c. Remove the **options nvme_core multipath=Y** line from the **/etc/modprobe.d/nvme_core.conf** file, if it is present.

   d. Reboot the system.

3. Make sure that DM Multipath is enabled:

```
# systemctl enable --now multipathd.service
```

4. Distribute I/O on all available paths. Add the following content in the **/etc/multipath.conf** file:

```
device {
  vendor "NVME"
  product ".*"
  path_grouping_policy    group_by_prio
}
```

NOTE

The **/sys/class/nvme-subsystem/nvme-subsys0/iopolicy** configuration file has no effect on the I/O distribution when DM Multipath manages the NVMe devices.

5. Reload the **multipathd** service to apply the configuration changes:

```
# multipath -r
```

6. Back up the **initramfs** file system:

```
# cp /boot/initramfs-$(uname -r).img \
    /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

7. Rebuild the **initramfs** file system:

```
# dracut --force --verbose
```

Verification

1. Check that your system recognizes the NVMe devices:

```
# nvme list

Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------
- ---------------- --------
/dev/nvme0n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme0n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme1n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme1n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme2n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme2n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme3n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme3n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
```

2. List all connected NVMe subsystems. Check that the command reports them as, for example, **nvme0n1** through **nvme3n2**, and *not* as, for example, **nvme0c0n1** through **nvme0c3n1**:

```
# nvme list-subsys

nvme-subsys0 - NQN=testnqn
\
 +- nvme0 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme1 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
 +- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

```
# multipath -ll
```

```
mpathae (uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03) dm-36 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 0:1:1:1  nvme0n1 259:0   active ready running
  |- 1:2:1:1  nvme1n1 259:2   active ready running
  |- 2:3:1:1  nvme2n1 259:4   active ready running
  `- 3:4:1:1  nvme3n1 259:6   active ready running

mpathaf (uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22) dm-39 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 0:1:2:2  nvme0n2 259:1   active ready running
  |- 1:2:2:2  nvme1n2 259:3   active ready running
  |- 2:3:2:2  nvme2n2 259:5   active ready running
  `- 3:4:2:2  nvme3n2 259:7   active ready running
```

Additional resources

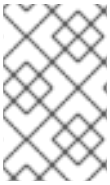
- Configuring kernel command-line parameters
- Configuring DM Multipath.

CHAPTER 5. MODIFYING THE DM-MULTIPATH CONFIGURATION FILE

By default, DM Multipath provides configuration values for the most common uses of multipathing. In addition, DM Multipath includes support for the most common storage arrays that themselves support DM Multipath. You can override the default configuration values for DM Multipath by editing the **/etc/multipath.conf** configuration file. If necessary, you can also add an unsupported by default storage array to the configuration file.

For information on the default configuration values, including supported devices, run either of the following commands:

```
# multipathd show config
# multipath -t
```



NOTE

If you run multipath from the **initramfs** file system and you make any changes to the multipath configuration files, you must rebuild the **initramfs** file system for the changes to take effect

In the multipath configuration file, you need to specify only the sections that you need for your configuration, or that you wish to change from the default values. If there are sections of the file that are not relevant to your environment or for which you do not need to override the default values, you can leave them commented out, as they are in the initial file.

The configuration file allows regular expression description syntax.

5.1. CONFIGURATION FILE OVERVIEW

The multipath configuration file is divided into the following sections:

blacklist

Listing of specific devices that will not be considered for multipath.

blacklist_exceptions

Listing of multipath devices that would otherwise be ignored according to the parameters of the **blacklist** section.

defaults

General default settings for DM Multipath.

multipaths

Settings for the characteristics of individual multipath devices. These values overwrite what is specified in the **overrides**, **devices**, and **defaults** sections of the configuration file.

devices

Settings for the individual storage controllers. These values overwrite what is specified in the **defaults** section of the configuration file. If you are using a storage array that is not supported by default, you may need to create a **devices** subsection for your array.

overrides

Settings that are applied to all devices. These values overwrite what is specified in the **devices** and **defaults** sections of the configuration file.

When the system determines the attributes of a multipath device, it checks the settings of the separate sections from the **multipath.conf** file in the following order:

1. **multipaths** section
2. **overrides** section
3. **devices** section
4. **defaults** section

5.2. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT

The **recovery_tmo sysfs** option controls the timeout for a particular iSCSI device. The following options globally override the **recovery_tmo** values:

- The **replacement_timeout** configuration option globally overrides the **recovery_tmo** value for all iSCSI devices.
- For all iSCSI devices that are managed by DM Multipath, the **fast_io_fail_tmo** option in DM Multipath globally overrides the **recovery_tmo** value.
  The **fast_io_fail_tmo** option in DM Multipath also overrides the **fast_io_fail_tmo** option in Fibre Channel devices.

The DM Multipath **fast_io_fail_tmo** option takes precedence over **replacement_timeout**. Red Hat does not recommend using **replacement_timeout** to override **recovery_tmo** in devices managed by DM Multipath because DM Multipath always resets **recovery_tmo**, when the **multipathd** service reloads.

5.3. MODIFYING MULTIPATH CONFIGURATION FILE DEFAULTS

The **/etc/multipath.conf** configuration file includes a **defaults** section that sets the **user_friendly_names** parameter to **yes**, as follows.

```
defaults {
        user_friendly_names yes
}
```

This overwrites the default value of the **user_friendly_names** parameter. The default values that are set in the defaults section on the **multipath.conf file**, are used by DM Multipath unless they are overwritten by the attributes specified in the devices, multipath, or overrides sections of the **multipath.conf** file.

Procedure

1. View the **/etc/multipath.conf** configuration file, which includes a template of configuration defaults:

```
#defaults {
#       polling_interval        10
#       path_selector           "round-robin 0"
#       path_grouping_policy    multibus
#       uid_attribute           ID_SERIAL
#       prio                    alua
#       path_checker            readsector0
#       rr_min_io               100
```

```
#       max_fds              8192
#       rr_weight            priorities
#       failback             immediate
#       no_path_retry        fail
#       user_friendly_names  yes
#}
```

2. Overwrite the default value for any of the configuration parameters. You can copy the relevant line from this template into the **defaults** section and uncomment it.
   For example, to overwrite the **path_grouping_policy** parameter to **multibus** instead of the default value of **failover**, copy the appropriate line from the template to the initial defaults section of the configuration file, and uncomment it, as follows:

```
defaults {
        user_friendly_names     yes
        path_grouping_policy    multibus
}
```

3. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

```
# multipath -t > /dev/null
```

   - To display the new configuration with the changes added, run:

```
# multipath -t
```

4. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

```
# service multipathd reload
```

Additional resources

- **multipath.conf(5)** and **multipathd(8)** man pages

5.4. MODIFYING MULTIPATH SETTINGS FOR SPECIFIC DEVICES

In the **multipaths** section of the **multipath.conf** configuration file, you can add configurations that are specific to an individual multipath device, referenced by the mandatory WWID parameter.

These defaults are used by DM Multipath and override attributes set in the **overrides**, **defaults**, and **devices** sections of the **multipath.conf** file. There can be any number of multipath subsections in the **multipaths** section.

Procedure

1. Modify the **multipaths** section for specific multipath device. The following example shows multipath attributes specified in the configuration file for two specific multipath devices:

- The first device has a WWID of **3600508b4000156d70001200000b0000** and a symbolic name of **yellow**.
- The second multipath device in the example has a WWID of **1DEC_321816758474** and a symbolic name of **red**.

In this example, the **rr_weight** attribute is set to **priorities**.

```
multipaths {
        multipath {
                wwid                    3600508b4000156d70001200000b0000
                alias                   yellow
                path_grouping_policy    multibus
                path_selector           "round-robin 0"
                failback                manual
                rr_weight               priorities
                no_path_retry           5
        }
        multipath {
                wwid                    1DEC_321816758474
                alias                   red
                rr_weight               priorities
        }
}
```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:
   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

Additional resources

- **multipath.conf(5)** man page

5.5. MODIFYING MULTIPATH SETTINGS FOR STORAGE CONTROLLERS

The **devices** section of the **multipath.conf** configuration file sets attributes for individual storage devices. These attributes are used by DM Multipath unless they are overwritten by the attributes specified in the **multipaths** or **overrides** sections of the **multipath.conf** file for paths that contain the device. These attributes override the attributes set in the **defaults** section of the **multipath.conf** file.

Procedure

1. View the information on the default configuration value, including supported devices:

```
# multipathd show config
# multipath -t
```

   Many devices that support multipathing are included by default in a multipath configuration.

2. Optional: If you need to modify the default configuration values, you can overwrite the default values by including an entry in the configuration file for the device that overwrites those values. You can copy the device configuration defaults for the device that the **multipathd show config** command displays and override the values that you want to change.

3. Add a device that is not configured automatically by default to the **devices** section of the configuration file by setting the **vendor** and **product** parameters. Find these values by opening the **/sys/block/*device_name*/device/vendor** and **/sys/block/*device_name*/device/model** files where *device_name* is the device to be multipathed, as mentioned in the following example:

```
# cat /sys/block/sda/device/vendor
WINSYS
# cat /sys/block/sda/device/model
SF2372
```

4. Optional: Specify the additional parameters depending on your specific device:

   active/active device

   Usually there is no need to set additional parameters in this case. If required, you might set **path_grouping_policy** to **multibus**. Other parameters you may need to set are **no_path_retry** and **rr_min_io**.

   active/passive device

   If it automatically switches paths with I/O to the passive path, you need to change the checker function to one that does not send I/O to the path to test if it is working, otherwise, your device will keep failing over. This means that you have set the **path_checker** to **tur**, which works for all SCSI devices that support the Test Unit Ready command, which most do.

   If the device needs a special command to switch paths, then configuring this device for multipath requires a hardware handler kernel module. The current available hardware handler is **emc**. If this is not sufficient for your device, you might not be able to configure the device for multipath.

   The following example shows a **device** entry in the multipath configuration file:

```
# }
# device {
#  vendor   "COMPAQ  "
#  product   "MSA1000         "
#  path_grouping_policy multibus
#  path_checker  tur
#  rr_weight  priorities
# }
#}
```

5. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

6. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

Additional resources

- **multipath.conf(5)** and **multipathd(8)** man pages

5.6. SETTING MULTIPATH VALUES FOR ALL DEVICES

Using the **overrides** section of the **multipath.conf** configuration file, you can set a configuration value for all of your devices. This section supports all attributes that are supported by both the **devices** and **defaults** section of the **multipath.conf** configuration file, which is all of the **devices** section attributes except **vendor**, **product**, and **revision**.

These attributes are used by DM Multipath for all devices unless they are overwritten by the attributes specified in the **multipaths** section of the **multipath.conf** file for paths that contain the device. These attributes override the attributes set in the **devices** and **defaults** sections of the **multipath.conf** file.

Procedure

1. Override device specific settings. For example, you might want all devices to set **no_path_retry** to **fail**. Use the following command to turn off queueing, when all paths have failed. This overrides any device specific settings.

   ```
   overrides {
           no_path_retry fail
   }
   ```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

```
# service multipathd reload
```

Additional resources

- **multipath.conf(5)** man page

CHAPTER 6. PREVENTING DEVICES FROM MULTIPATHING

You can configure DM Multipath to ignore selected devices when it configures multipath devices. DM Multipath does not group these ignored devices into a multipath device.

6.1. CONDITIONS WHEN DM MULTIPATH CREATES A MULTIPATH DEVICE FOR A PATH

DM Multipath has a set of default rules to determine whether to create a multipath device for a path or whether to ignore the path. You can configure the behavior.

If the **find_multipaths** configuration parameter is set to **off**, multipath always tries to create a multipath device for every path that is not explicitly disabled. If the **find_multipaths** configuration parameter is set to **on**, then multipath creates a device, only if one of following conditions is met:

- There are at least two paths with the same World-Wide Identification (WWID) that are not disabled.
- You manually force the creation of the device by specifying a device with the **multipath** command.
- A path has the same WWID as a multipath device that was previously created even if that multipath device does not currently exist. Whenever a multipath device is created, multipath remembers the WWID of the device so that it automatically creates the device again as soon as it sees a path with that WWID. This allows you to have multipath automatically choose the correct paths to make into multipath devices, without having to disable multipathing on other devices.

If you have previously created a multipath device without using the **find_multipaths** parameter and then you later set the parameter to **on**, you might need to remove the WWIDs of any device you do not want created as a multipath device from the **/etc/multipath/wwids** file. The following example shows a sample **/etc/multipath/wwids** file. The WWIDs are enclosed by slashes (/):

```
# Multipath wwids, Version : 1.0
# NOTE: This file is automatically maintained by multipath and multipathd.
# You should not need to edit this file in normal circumstances.
#
# Valid WWIDs:
/3600d0230000000000e13955cc3757802/
/3600d0230000000000e13955cc3757801/
/3600d0230000000000e13955cc3757800/
/3600d02300069c9ce09d41c31f29d4c00/
/SWINSYS  SF2372        0E13955CC3757802/
/3600d0230000000000e13955cc3757803/
```

In addition to **on** and **off**, you can also set **find_multipaths** to the following values:

strict

Multipath never accepts paths that have not previously been multipathed and are therefore not in the **/etc/multipath/wwids** file.

smart

Multipath always accepts non-disabled devices in **udev** as soon as they appear. If **multipathd** does not create the device within a timeout set with the **find_multipaths_timeout** parameter, it will release its claim on the device.

The built-in default value of **find_multipaths** is **off**. The default **multipath.conf** file created by **mpathconf**, however, will set the value of **find_multipaths** to **on**.

When the **find_multipaths** parameter is set to **on**, disable multipathing only on the devices with multiple paths that you do not want to be multipathed. Because of this, it will generally not be necessary to disable multipathing on devices.

If you add a previously created multipath device to **blacklist**, removing the WWID of that device from the **/etc/multipath/wwids** file by using the **-w** option can help avoid issues with other programs. For example, to remove the device **/dev/sdb** with WWID **3600d0230000000000e13954ed5f89300** from the **/etc/multipath/wwids** file, you can use either of the following methods.

- Removing a multipath device by using the device name.

```
#multipath -w /dev/sdb
wwid '3600d0230000000000e13954ed5f89300' removed
```

- Removing a multipath device by using the WWID of the device.

```
#multipath -w 3600d0230000000000e13954ed5f89300
wwid '3600d0230000000000e13954ed5f89300' removed
```

You can also use the **-W** option to update the **/etc/multipath/wwids** file. This would reset the **/etc/multipath/wwids** file to only include the WWIDs of the current multipath devices. To reset the file, run the following:

```
#multipath -W
successfully reset wwids
```

Additional resources

- **multipath.conf(5)** man page

6.2. CRITERIA FOR DISABLING MULTIPATHING ON CERTAIN DEVICES

You can disable multipathing on devices by any of the following criteria:

- WWID
- device name
- device type
- property
- protocol

For every device, DM Multipath evaluates these criteria in the following order:

1. **property**
2. **devnode**
3. **device**

4. **protocol**
5. **wwid**

If a device turns out to be disabled by any of the mentioned criteria, DM Multipath excludes it from handling by **multipathd**, and does not evaluate the later criteria. For each criteria, the exception list takes precedence over the list of disabled devices, if a device matches both.

NOTE

By default, a variety of device types are disabled, even after you comment out the initial **blacklist** section of the configuration file.

Additional resources

- Adding exceptions for devices with disabled multipathing

6.3. DISABLING MULTIPATHING BY WWID

You can disable multipathing on individual devices by their World-Wide Identification (WWID).

Procedure

1. Disable devices in the **/etc/multipath.conf** configuration file using the **wwid** entry.
   The following example shows the lines in the DM Multipath configuration file that disable a device with a WWID of **26353900f02796769**:

```
blacklist {
        wwid 26353900f02796769
}
```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:
   - To display any configuration errors, run:

```
# multipath -t > /dev/null
```

   - To display the new configuration with the changes added, run:

```
# multipath -t
```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

```
# service multipathd reload
```

6.4. DISABLING MULTIPATHING BY DEVICE NAME

You can disable multipathing on device types by device name, so that DM Multipath will not group them into a multipath device.

Procedure

1. Disable devices in the **/etc/multipath.conf** configuration file using the **devnode** entry. The following example shows the lines in the DM Multipath configuration file that disable all SCSI devices, because it disables all **sd*** devices as well:

```
blacklist {
        devnode "^sd[a-z]"
}
```

   You can use a **devnode** entry to disable individual devices rather than all devices of a specific type. However, this is not recommended because unless it is statically mapped by **udev** rules, there is no guarantee that a specific device will have the same name on reboot. For example, a device name could change from **/dev/sda** to **/dev/sdb** on reboot.

   By default, DM Multipath disables all devices that are not SCSI, NVMe, or DASD, using the following **devnode** entry:

```
blacklist {
        devnode "!^(sd[a-z]|dasd[a-z]|nvme[0-9])"
}
```

   The devices that this entry disables do not generally support DM Multipath.

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

```
# multipath -t > /dev/null
```

   - To display the new configuration with the changes added, run:

```
# multipath -t
```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

```
# service multipathd reload
```

Additional resources

- Adding exceptions for devices with disabled multipathing

6.5. DISABLING MULTIPATHING BY DEVICE TYPE

You can disable multipathing on devices by using the device section.

Procedure

1. Disable devices in the **/etc/multipath.conf** configuration file using the **device** section. The following example disables multipathing on all IBM DS4200 and HP devices:

```
blacklist {
        device {
                vendor  "IBM"
                product "3S42"       #DS4200 Product 10
        }
        device {
                vendor  "HP"
                product ".*"
        }
}
```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

6.6. DISABLING MULTIPATHING BY UDEV PROPERTY

You can disable multipathing on devices by their **udev** property parameter.

Procedure

1. Disable devices in the **/etc/multipath.conf** configuration file using the **property** parameter. This parameter is a regular expression string that matches against the **udev** environment variable name for the devices.
   The following example disables multipathing on all devices with the **udev** property **ID_ATA**:

   ```
   blacklist {
          property "ID_ATA"
   }
   ```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

```
# service multipathd reload
```

6.7. DISABLING MULTIPATHING BY DEVICE PROTOCOL

You can disable multipathing on devices by using device protocol.

Procedure

1. Optional: View the protocol that a path is using:

```
# multipathd show paths format "%d %P"
```

2. Disable devices in the **/etc/multipath.conf** configuration file using the **protocol** section. The following example disables multipathing on all devices with an undefined protocol or an unknown SCSI transport type:

```
blacklist {
        protocol "scsi:unspec"
        protocol "undef"
}
```

DM Multipath recognizes the following protocol strings:

- **scsi:fcp**
- **scsi:spi**
- **scsi:ssa**
- **scsi:sbp**
- **scsi:srp**
- **scsi:iscsi**
- **scsi:sas**
- **scsi:adt**
- **scsi:ata**
- **scsi:unspec**
- **ccw**
- **cciss**
- **nvme**
- **undef**

3. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

4. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

6.8. ADDING EXCEPTIONS FOR DEVICES WITH DISABLED MULTIPATHING

You can enable multipathing by adding exceptions on devices where multipathing is currently disabled.

Prerequisites

- Multipathing is disabled on certain devices.

Procedure

1. Enable multipathing on the devices using the **blacklist_exceptions** section of the **/etc/multipath.conf** configuration file.
   When specifying devices in the **blacklist_exceptions** section of the configuration file, you must specify the exceptions using the same criteria as they were specified in the **blacklist** section. For example, a WWID exception does not apply to devices disabled by a **devnode** entry, even if the disabled device is associated with that WWID. Similarly, **devnode** exceptions apply only to **devnode** entries, and **device** exceptions apply only to device entries.

   Example 6.1. An exception by WWID

   If you have a large number of devices and want to multipath only one of them with the WWID of **3600d0230000000000e13955cc3757803**, instead of individually disabling each of the devices except the one you want, you could disable all of them, and then enable only the one you want by adding the following lines to the **/etc/multipath.conf** file:

   ```
   blacklist {
         wwid ".*"
   }

   blacklist_exceptions {
         wwid "3600d0230000000000e13955cc3757803"
   }
   ```

   Alternatively, you could use an exclamation mark (**!**) to invert the **blacklist** entry, which disables all devices except the specified WWID:

```
blacklist {
        wwid "!3600d0230000000000e13955cc3757803"
}
```

Example 6.2. An exception by udev property

The **property** parameter works differently than the other **blacklist_exception** parameters. The value of the **property** parameter must match the name of a variable in the **udev** database. Otherwise, the device is disabled. Using this parameter, you can disable multipathing on certain SCSI devices, such as USB sticks and local hard drives.

To enable multipathing only on SCSI devices that could reasonably be multipathed, set this parameter to **(SCSI_IDENT_|ID_WWN)** as in the following example:

```
blacklist_exceptions {
        property "(SCSI_IDENT_|ID_WWN)"
}
```

2. Validate the **/etc/multipath.conf** file after modifying the multipath configuration file by running one of the following commands:

   - To display any configuration errors, run:

     ```
     # multipath -t > /dev/null
     ```

   - To display the new configuration with the changes added, run:

     ```
     # multipath -t
     ```

3. Reload the **/etc/multipath.conf** file and reconfigure the **multipathd** daemon for changes to take effect:

   ```
   # service multipathd reload
   ```

CHAPTER 7. MANAGING MULTIPATHED VOLUMES

The following are a few commands provided by DM Multipath, which you can use to manage multipath volumes:

- **multipath**
- **dmsetup**
- **multipathd**

7.1. RESIZING AN ONLINE MULTIPATH DEVICE

If you need to resize an online multipath device, use the following procedure.

Procedure

1. Resize your physical device.
2. Execute the following command to find the paths to the logical unit number (LUN):

```
# multipath -l
```

3. Resize your paths. For SCSI devices, writing a 1 to the **rescan** file for the device causes the SCSI driver to rescan, as in the following command:

```
# echo 1 > /sys/block/path_device/device/rescan
```

Ensure that you run this command for each of the path devices. For example, if your path devices are **sda**, **sdb**, **sde**, and **sdf**, you would run the following commands:

```
# echo 1 > /sys/block/sda/device/rescan
# echo 1 > /sys/block/sdb/device/rescan
# echo 1 > /sys/block/sde/device/rescan
# echo 1 > /sys/block/sdf/device/rescan
```

4. Resize your multipath device:

```
# multipathd resize map multipath_device
```

5. Resize the file system (assuming no LVM or DOS partitions are used):

```
# resize2fs /dev/mapper/mpatha
```

7.2. MOVING A ROOT FILE SYSTEM FROM A SINGLE PATH DEVICE TO A MULTIPATH DEVICE

If you have installed your system on a single-path device and later add another path to the root file system, you will need to move your root file system to a multipathed device. See the following procedure for moving from a single-path to a multipathed device.

Prerequisites

- You have installed the **device-mapper-multipath** package.

Procedure

1. Create the **/etc/multipath.conf** configuration file, load the multipath module, and enable the **multipathd systemd** service:

```
# yum install device-mapper-multipath
```

Procedure

1. Execute the following command to create the **/etc/multipath.conf** configuration file, load the multipath module, and set **chkconfig** for the **multipathd** to **on**:

2. If the **find_multipaths** configuration parameter is not set to **yes**, edit the **blacklist** and **blacklist_exceptions** sections of the **/etc/multipath.conf** file, as described in Preventing devices from multipathing.

3. In order for multipath to build a multipath device on top of the root device as soon as it is discovered, enter the following command. This command also ensures that **find_multipaths** allows the device, even if it only has one path.

```
# multipath -a root_devname
```

For example, if the root device is **/dev/sdb**, enter the following command.

```
# multipath -a /dev/sdb
wwid '3600d02300069c9ce09d41c4ac9c53200' added
```

4. Confirm that your configuration file is set up correctly by executing the **multipath** command and search the output for a line of the following format. This indicates that the command failed to create the multipath device.

```
date  wwid: ignoring map
```

For example, if the WWID of the device is *3600d02300069c9ce09d41c4ac9c53200*, you would see a line in the output such as the following:

```
# multipath
Oct 21 09:37:19 | 3600d02300069c9ce09d41c4ac9c53200: ignoring map
```

5. Rebuild the **initramfs** file system with **multipath**:

```
# dracut --force -H --add multipath
```

6. Shut the machine down.

7. Boot the machine.

8. Make the other paths visible to the machine.

Verification steps

- Check whether the multipath device is created by running the following command:

```
# multipath -l | grep 3600d02300069c9ce09d41c4ac9c53200
mpatha (3600d02300069c9ce09d41c4ac9c53200) dm-0 3PARdata,VV
```

7.3. MOVING A SWAP FILE SYSTEM FROM A SINGLE PATH DEVICE TO A MULTIPATH DEVICE

By default, swap devices are set up as logical volumes. This does not require any special procedure for configuring them as multipath devices as long as you set up multipathing on the physical volumes that constitute the logical volume group. If your swap device is not an LVM volume, however, and it is mounted by device name, you might need to edit the **/etc/fstab** file to switch to the appropriate multipath device name.

Procedure

1. Add the WWID of the device to the **/etc/multipath/wwids** file:

```
# multipath -a swap_devname
```

   For example, if the root device is **/dev/sdb**, enter the following command.

```
# multipath -a /dev/sdb
wwid '3600d02300069c9ce09d41c4ac9c53200' added
```

2. Confirm that your configuration file is set up correctly by executing the **multipath** command and search the output for a line of the following format:

```
date  wwid: ignoring map
```

   This indicates that the command failed to create the multipath device.

   For example, if the WWID of the device is 3600d02300069c9ce09d41c4ac9c53200, you would see a line in the output such as the following:

```
# multipath
Oct 21 09:37:19 | 3600d02300069c9ce09d41c4ac9c53200: ignoring map
```

3. Set up an alias for the swap device in the **/etc/multipath.conf** file:

```
multipaths {
    multipath {
        wwid WWID_of_swap_device
        alias swapdev
    }
}
```

4. Edit the **/etc/fstab** file and replace the old device path to the root device with the multipath device.
   For example, if you had the following entry in the **/etc/fstab** file:

```
/dev/sdb2 swap                swap    defaults        0 0
```

Change the entry to the following:

```
/dev/mapper/swapdev swap          swap    defaults        0 0
```

5. Rebuild the initramfs file system with multipath:

```
# dracut --force -H --add multipath
```

6. Shut the machine down.

7. Boot the machine.

8. Make the other paths visible to the machine.

Verification steps

- Verify if the the swap device is on the multipath device:

```
# swapon -s
```

For example:

```
# swapon -s

Filename                Type        Size Used   Priority
/dev/dm-3               partition   4169724 0   -2
```

The file name should match the multipath swap device.

```
# readlink -f /dev/mapper/swapdev
/dev/dm-3
```

7.4. DETERMINING DEVICE MAPPER ENTRIES WITH THE DMSETUP COMMAND

You can use the **dmsetup** command to find out which device mapper entries match the multipathed devices.

Procedure

- Display all the device mapper devices and their major and minor numbers. The minor numbers determine the name of the dm device. For example, a minor number of 3 corresponds to the multipathed device **/dev/dm-3**.

```
# dmsetup ls
mpathd  (253:4)
mpathep1        (253:12)
mpathfp1        (253:11)
mpathb  (253:3)
mpathgp1        (253:14)
mpathhp1        (253:13)
mpatha  (253:2)
```

```
mpathh  (253:9)
mpathg  (253:8)
VolGroup00-LogVol01     (253:1)
mpathf  (253:7)
VolGroup00-LogVol00     (253:0)
mpathe  (253:6)
mpathbp1        (253:10)
mpathd  (253:5)
```

7.5. ADMINISTERING THE MULTIPATHD DAEMON

The **multipathd** commands can be used to administer the **multipathd** daemon.

Procedure

- View the standard default format for the output of the **multipathd show maps** command:

```
# multipathd show maps
name sysfs uuid
mpathc dm-0 360a98000324669436c2b45666c567942
```

- Some **multipathd** commands include a **format** option followed by a wildcard. Display a list of available wildcards with the following command:

```
# multipathd show wildcards
```

- Display the multipath devices that **multipathd** is monitoring, using a format string with multipath wildcards, in regular and raw format:

```
list|show maps|multipaths format $format
list|show maps|multipaths raw format $format
```

The **multipathd** command supports format commands that show the status of multipath devices and paths in "raw" format versions. In raw format, no headers are printed and the fields are not padded to align the columns with the headers. Instead, the fields print exactly as specified in the format string. This output can then be more easily used for scripting. You can display the wildcards used in the format string with the **multipathd show wildcards** command.

- Display the paths that **multipathd** is monitoring, using a format string with multipath wildcards, in regular and raw format:

```
list|show paths format $format
list|show paths raw format $format
```

- Display the difference between the non-raw and raw formats for the **multipathd show maps**. Note that in **raw** format there are no headers and only a single space between the columns:

```
# multipathd show maps format "%n %w %d %s"
name   uuid                             sysfs vend/prod/rev
mpathc 360a98000324669436c2b45666c567942 dm-0  NETAPP,LUN

# multipathd show maps raw format "%n %w %d %s"
mpathc 360a98000324669436c2b45666c567942 dm-0 NETAPP,LUN
```

Additional resources

- **multipathd**(8) man page

CHAPTER 8. REMOVING STORAGE DEVICES

You can safely remove a storage device from a running system, which helps prevent system memory overload and data loss.

Prerequisites

- Before you remove a storage device, you must ensure that you have enough free system memory due to the increased system memory load during an I/O flush. Use the following commands to view the current memory load and free memory of the system:

```
# vmstat 1 100
# free
```

- Red Hat does not recommend removing a storage device on a system where:
  - Free memory is less than 5% of the total memory in more than 10 samples per 100.
  - Swapping is active (non-zero **si** and **so** columns in the **vmstat** command output).

8.1. SAFE REMOVAL OF STORAGE DEVICES

Safely removing a storage device from a running system requires a top-to-bottom approach. Start from the top layer, which typically is an application or a file system, and work towards the bottom layer, which is the physical device.

You can use storage devices in multiple ways, and they can have different virtual configurations on top of physical devices. For example, you can group multiple instances of a device into a multipath device, make it part of a RAID, or you can make it part of an LVM group. Additionally, devices can be accessed via a file system, or they can be accessed directly such as a "raw" device.

While using the top-to-bottom approach, you must ensure that:

- the device that you want to remove is not in use
- all pending I/O to the device is flushed
- the operating system is not referencing the storage device

8.2. REMOVING A BLOCK DEVICE

You can safely remove a block device from a running system to help prevent system memory overload and data loss.

WARNING

Rescanning the SCSI bus or performing any other action that changes the state of the operating system, without following the procedure documented here can cause delays due to I/O timeouts, devices to be removed unexpectedly, or data loss.

Prerequisites

- If you want to remove a multipath device, and you are unable to access its path devices, disable queueing of the multipath device:

```
# multipathd disablequeueing map multipath-device
```

  This enables the I/O of the device to fail, allowing the applications that are using the device to shut down.

- Ensure that no other applications or services are using the device that you want to remove.
- Ensure that you back up the data from the device that you want to remove.

Procedure

1. Unmount any file systems that are mounted on the device using the **umount** command.
2. Remove the device from any MD RAID array or from any LVM volume that it belongs to. Depending on the device type, execute one of the following steps:
   - If the device is a member of an LVM group, and it is a multipath device:
     a. Move the data to another device:

```
# pvmove -b /dev/mapper/from-multipath-device /dev/mapper/to-multipath-device
```

     b. Remove the device from the volume group:

```
# vgreduce volume-group /dev/mapper/from-multipath-device
```

     c. Optional: Remove the LVM metadata from the physical device:

```
# pvremove /dev/mapper/from-multipath-device
```

   - If you are removing a multipath device, execute the following commands:
     a. View all the paths to the device:

```
# multipath -l
```

     The output of this command is required in a later step.

     b. Flush the I/O and remove the multipath device:

```
# multipath -f multipath-device
```

   - If the device is not configured as a multipath device, or if the device is configured as a multipath device and you have previously passed I/O to the individual paths, flush any outstanding I/O to all device paths that are used:

```
# blockdev --flushbufs device
```

This is important for devices accessed directly where the **umount** or **vgreduce** commands do not flush the I/O.

- If you are removing a SCSI device, execute the following commands:

  a. Remove any reference to the path-based name of the device, such as ***/dev/sd***, ***/dev/disk/by-path***, or the **major:minor** number, in applications, scripts, or utilities on the system. This ensures that different devices added in the future are not mistaken for the current device.

  b. Remove each path to the device from the SCSI subsystem:

  ```
  # echo 1 > /sys/block/device-name/device/delete
  ```

  where ***device-name*** is retrieved from the output of the **multipath -l** command, if the device was previously used as a multipath device.

3. Remove the physical device from a running system. Note that the I/O to other devices does not stop when you remove this device.

Additional resources

- The **multipath(8)**, **pvmove(8)**, **vgreduce(8)**, **blockdev(8)** and **umount(8)** man pages.

CHAPTER 9. TROUBLESHOOTING DM MULTIPATH

If you have trouble implementing a multipath configuration, there are a variety of issues you can check for. The following issues may result in a slow or non-functioning multipath configuration:

The multipath daemon is not running

If you find you have trouble implementing a multipath configuration, ensure that the **multipathd** daemon is running, as described in Setting up DM Multipath. The **multipathd** daemon must be running in order to use multipathed devices.

Issues with queue_if_no_path feature

If a multipath device is configured with the **features "1 queue_if_no_path"** option, then any process that issues I/O hangs until one or more paths are restored.

9.1. TROUBLESHOOTING ISSUES WITH QUEUE_IF_NO_PATH FEATURE

If a multipath device is configured with the **features "1 queue_if_no_path"** option, then any process that issues I/O hangs until one or more paths are restored. To avoid this, set the **no_path_retry N** parameter in the **/etc/multipath.conf** file, where *N* is the number of times the system should retry a path.

If you need to use the **features "1 queue_if_no_path"** option and you experience the issue noted here, you can disable the queueing policy at runtime for a particular LUN, for which all the paths are unavailable.

Procedure

- Disable queueing for a specific device:

```
# multipathd disablequeueing map device
```

- Disable queueing for all devices:

```
# multipathd disablequeueing maps
```

After you have disabled queueing for a device, it will remain disabled until **multipathd** is restarted or reloaded, or until you execute one of the following commands:

- Reset queueing to the previous value for a specific device:

```
# multipathd restorequeueing map device
```

- Reset queueing to the previous value for all devices:

```
# multipathd restorequeueing maps
```

9.2. TROUBLESHOOTING WITH THE MULTIPATHD INTERACTIVE CONSOLE

The **multipathd -k** command is an interactive interface to the **multipathd** daemon. Entering this command brings up an interactive multipath console. After executing this command, you can enter **help** to get a list of available commands and **Ctrl**+**D** to quit.

Use the **multipathd** interactive console to troubleshoot problems you might have with your system.

Procedure

- Display the multipath configuration, including the defaults, before exiting the console:

```
# multipathd -k
multipathd> show config
multipathd> Ctrl+D
```

- Ensure that multipath has picked up any changes to the **multipath.conf** file:

```
# multipathd -k
multipathd> reconfigure
multipathd> Ctrl+D
```

- Ensure that the path checker is working properly:

```
# multipathd -k
multipathd> show paths
multipathd> Ctrl+D
```

- You can also run a single **multipathd** interactive command directly from the command line, without starting the interactive console. For example, to check that multipath has picked up any changes to the **multipath.conf** file, run:

```
# multipathd reconfigure
```

CHAPTER 10. CONFIGURING MAXIMUM TIME FOR STORAGE ERROR RECOVERY WITH EH_DEADLINE

You can configure the maximum allowed time to recover failed SCSI devices. This configuration guarantees an I/O response time even when storage hardware becomes unresponsive due to a failure.

10.1. THE EH_DEADLINE PARAMETER

The SCSI error handling (EH) mechanism attempts to perform error recovery on failed SCSI devices. The SCSI host object **eh_deadline** parameter enables you to configure the maximum amount of time for the recovery. After the configured time expires, SCSI EH stops and resets the entire host bus adapter (HBA).

Using **eh_deadline** can reduce the time:

- to shut off a failed path,
- to switch a path, or
- to disable a RAID slice.

WARNING

When **eh_deadline** expires, SCSI EH resets the HBA, which affects all target paths on that HBA, not only the failing one. If some of the redundant paths are not available for other reasons, I/O errors might occur. Enable **eh_deadline** only if you have a fully redundant multipath configuration on all targets.

The value of the **eh_deadline** parameter is specified in seconds. The default setting is **off**, which disables the time limit and allows all of the error recovery to take place.

Scenarios when eh_deadline is useful

In most scenarios, you do not need to enable **eh_deadline**. Using **eh_deadline** can be useful in certain specific scenarios. For example if a link loss occurs between a Fibre Channel (FC) switch and a target port, and the HBA does not receive Registered State Change Notifications (RSCNs). In such a case, I/O requests and error recovery commands all time out rather than encounter an error. Setting **eh_deadline** in this environment puts an upper limit on the recovery time. That enables the failed I/O to be retried on another available path by DM Multipath.

Under the following conditions, the **eh_deadline** parameter provides no additional benefit, because the I/O and error recovery commands fail immediately, which enables DM Multipath to retry:

- If RSCNs are enabled
- If the HBA does not register the link becoming unavailable

10.2. SETTING THE EH_DEADLINE PARAMETER

This procedure configures the value of the **eh_deadline** parameter to limit the maximum SCSI recovery time.

Procedure

- You can configure **eh_deadline** using either of the following methods:
  - **defaults** section of the **multipath.conf** file
    From the defaults section of the **multipath.conf** file, set the **eh_deadline** parameter to the required number of seconds:

    ```
    # eh_deadline 300
    ```

    NOTE

    From RHEL 8.4, setting the **eh_deadline** parameter using the defaults section of the **multipath.conf** file is the preferred method.

    To turn off the **eh_deadline** parameter with this method, set **eh_deadline** to **off**.

  - **sysfs**
    Write the number of seconds into the **/sys/class/scsi_host/host<host-number>/eh_deadline** files. For example, to set the **eh_deadline** parameter through **sysfs** on SCSI host 6:

    ```
    # echo 300 > /sys/class/scsi_host/host6/eh_deadline
    ```

    To turn off the **eh_deadline** parameter with this method, use echo **off**.

  - Kernel parameter
    Set a default value for all SCSI HBAs using the **scsi_mod.eh_deadline** kernel parameter.

    ```
    # echo 300 > /sys/module/scsi_mod/parameters/eh_deadline
    ```

    To turn off the **eh_deadline** parameter with this method, use echo **-1**.

Additional resources

- How to set eh_deadline and eh_timeout persistently, using a udev rule