Red Hat Enterprise Linux 7

DM Multipath

Configuring and managing Device Mapper Multipath

Last Updated: 2019-08-21

Red Hat Enterprise Linux 7 DM Multipath

Configuring and managing Device Mapper Multipath

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

Legal Notice

Abstract

This book provides information on using the Device Mapper Multipath feature on Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. DEVICE MAPPER MULTIPATHING

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

This chapter provides a summary of the features of DM-Multipath that were added subsequent to the initial release of Red Hat Enterprise Linux 7. Following that, this chapter provides a high level overview of DM Multipath and its components, as well as an overview of DM-Multipath setup.

1.1. NEW AND CHANGED FEATURES

This section lists features of the DM Multipath that are new since the initial release of Red Hat Enterprise Linux 7.

1.1.1. New and Changed Features for Red Hat Enterprise Linux 7.1

Red Hat Enterprise Linux 7.1 includes the following documentation and feature updates and changes.

- Table 5.1, "Useful **multipath** Command Options". now includes entries for the -w and -W options of the **multipath** command, which allow you to better manage the wwids file.
- Additional options for the **values** argument of the **features** parameter in the **multipath.conf** file are documented in Chapter 4, *The DM Multipath Configuration File* .
- Table 4.1, "Multipath Configuration Defaults". includes an entry for the **force_sync** parameter, which prevents path checkers from running in async mode when set to "yes".

In addition, small technical corrections and clarifications have been made throughout the document.

1.1.2. New and Changed Features for Red Hat Enterprise Linux 7.2

Red Hat Enterprise Linux 7.2 includes the following documentation and feature updates and changes.

- This document includes a new section, Section 5.1, "Automatic Configuration File Generation with Multipath Helper". The Multipath Helper application gives you options to create multipath configurations with custom aliases, device blacklists, and settings for the characteristics of individual multipath devices.
- The **defaults** section of the **multipath.conf** configuration file supports the new **config_dir**, **new_bindings_in_boot**, **ignore_new_boot_devs**, **retrigger_tries**, and **retrigger_delays** parameters. The **defaults** section of the **multipath.conf** file is documented in Table 4.1, "Multipath Configuration Defaults".
- The **defaults**, **devices**, and **multipaths** sections of the **multipath.conf** configuration file now support the **delay_watch_checks** and **delay_wait_checks** configuration parameters. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File*.

In addition, small technical corrections and clarifications have been made throughout the document.

1.1.3. New and Changed Features for Red Hat Enterprise Linux 7.3

Red Hat Enterprise Linux 7.3 includes the following documentation and feature updates and changes.

- The **multipathd** command supports new 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. For information on the **multipathd** commands, see Section 5.12, "The multipathd Commands".

- As of Red Hat Enterprise Linux 7.3, if you specify **prio "alua exclusive_pref_bit"** in your device configuration, multipath will create a path group that contains only the path with the **pref** bit set and will give that path group the highest priority. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File* .

- The **defaults**, **devices**, and **multipaths** sections of the **multipath.conf** configuration file now support the **skip_kpartx** configuration parameter. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File* .

In addition, small technical corrections and clarifications have been made throughout the document.

1.1.4. New and Changed Features for Red Hat Enterprise Linux 7.4

Red Hat Enterprise Linux 7.4 includes the following documentation and feature updates and changes.

- The **defaults**, **devices**, and **multipaths** sections of the **multipath.conf** configuration file support the **max_sectors_kb** configuration parameter. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File* .

- The **defaults** and **devices** sections of the **multipath.conf** configuration file support the **detect_path_checker** configuration parameter. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File* .

- The **defaults** section of the **multipath.conf** configuration file supports the **remove_retries** and **detect_path_checker** parameters. The **defaults** section of the **multipath.conf** file is documented in Table 4.1, "Multipath Configuration Defaults".

1.1.5. New and Changed Features for Red Hat Enterprise Linux 7.5

Red Hat Enterprise Linux 7.5 includes the following documentation and feature updates and changes.

- As of Red Hat Enterprise Linux 7.5, the **blacklist** and **blacklist_exceptions** section of the **multipath.conf** configuration file support the **property** parameter. For information on the **property** parameter, see Section 4.2.6, "Blacklist Exceptions".

- the **defaults** and **multipaths** sections of the **multipath.conf** file now support a value of **file** for the **reservation_key** parameter. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File* .

- The **defaults** section of the **multipath.conf** configuration file supports the **prkeys_file** parameter. The **defaults** section of the **multipath.conf** file is documented in Table 4.1, "Multipath Configuration Defaults".

1.1.6. New and Changed Features for Red Hat Enterprise Linux 7.6

Red Hat Enterprise Linux 7.6 includes the following documentation and feature updates and changes.

- As of Red Hat Enterprise Linux 7.6, you can specify the protocol for a device to be excluded from multipathing in the **blacklist** section of the configuration file with a **protocol** section. For information on blacklisting by device protocol, see Section 4.2.5, "Blacklisting By Device

Protocol (Red Hat Enterprise Linux 7.6 and Later)".

- The **defaults** and **devices** sections of the **multipath.conf** configuration file support the **all_tg_pt** parameter. For information on the configuration parameters, see Chapter 4, *The DM Multipath Configuration File*.

1.2. OVERVIEW OF DM MULTIPATH

DM Multipath can be used to provide:

- 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 (the cable, switch, or controller) fails, DM Multipath switches to an alternate path.

- Improved Performance

  DM Multipath can be configured in 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.

Figure 1.1, "Active/Passive Multipath Configuration with One RAID Device" shows an active/passive configuration with two I/O paths from the server to a RAID device. There are 2 HBAs on the server, 2 SAN switches, and 2 RAID controllers.

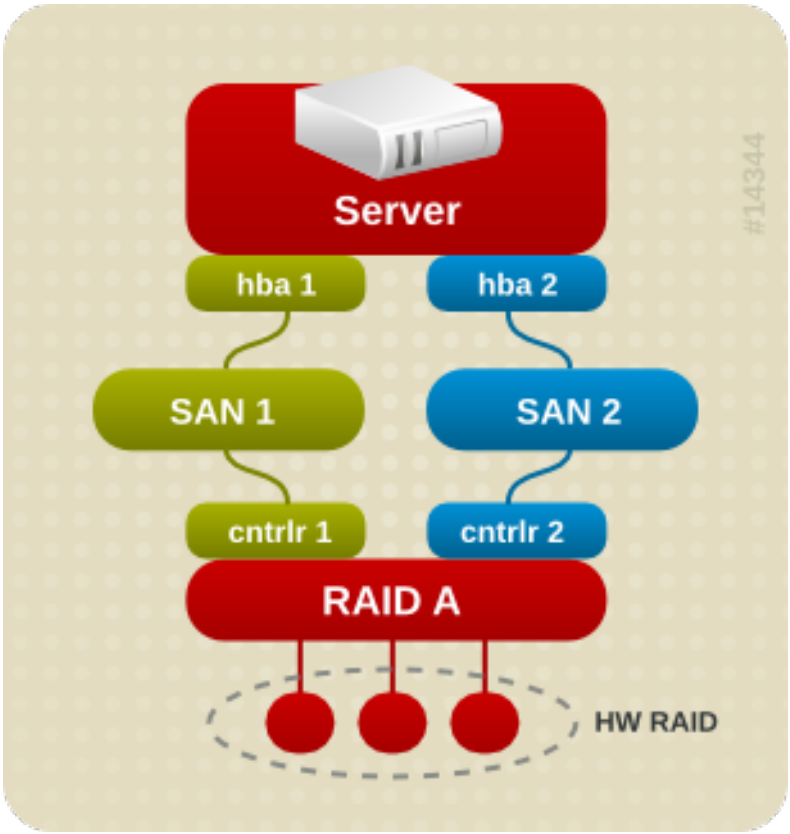


Figure 1.1. Active/Passive Multipath Configuration with One RAID Device

In this configuration, there is one I/O path that goes through hba1, SAN1, and controller 1 and a second I/O path that goes through hba2, SAN2, and controller2. There are many points of possible failure in this configuration:

- HBA failure
- FC cable failure
- SAN switch failure
- Array controller port failure

With DM Multipath configured, a failure at any of these points will cause DM Multipath to switch to the alternate I/O path.

Figure 1.2, "Active/Passive Multipath Configuration with Two RAID Devices" shows a more complex active/passive configuration with 2 HBAs on the server, 2 SAN switches, and 2 RAID devices with 2 RAID controllers each.

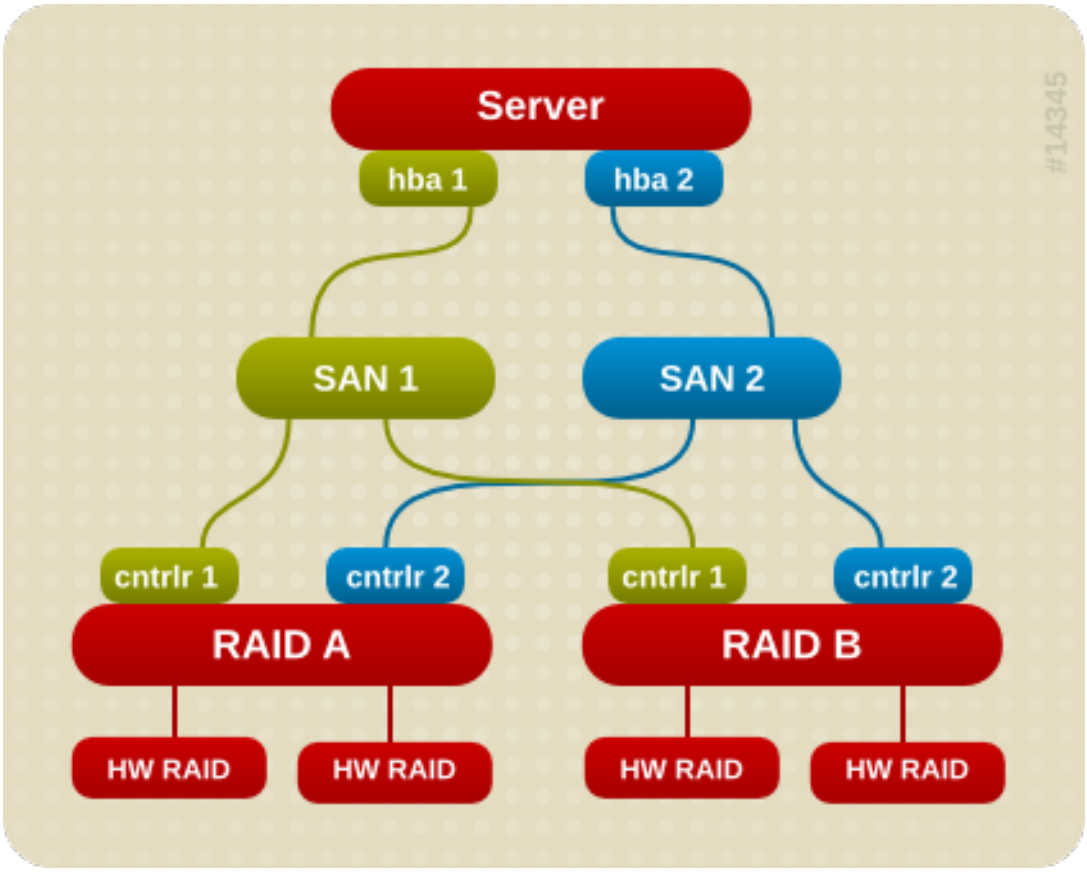


Figure 1.2. Active/Passive Multipath Configuration with Two RAID Devices

In the example shown in Figure 1.2, "Active/Passive Multipath Configuration with Two RAID Devices", there are two I/O paths to each RAID device (just as there are in the example shown in Figure 1.1, "Active/Passive Multipath Configuration with One RAID Device"). With DM Multipath configured, a failure at any of the points of the I/O path to either of the RAID devices will cause DM Multipath to switch to the alternate I/O path for that device.

Figure 1.3, "Active/Active Multipath Configuration with One RAID Device" shows an active/active configuration with 2 HBAs on the server, 1 SAN switch, and 2 RAID controllers. There are four I/O paths from the server to a storage device:

- hba1 to controller1
- hba1 to controller2
- hba2 to controller1

- hba2 to controller2

In this configuration, I/O can be spread among those four paths.

Figure 1.3. Active/Active Multipath Configuration with One RAID Device

1.3. STORAGE ARRAY SUPPORT

By default, DM Multipath includes support for the most common storage arrays that themselves support DM Multipath. For information on the default configuration values, including supported devices, run either of the following commands.

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

If your storage array supports DM Multipath and is not configured by default, you may need to add it to the DM Multipath configuration file, **multipath.conf**. For information on the DM Multipath configuration file, see Chapter 4, *The DM Multipath Configuration File* .

Some storage arrays require special handling of I/O errors and path switching. These require separate hardware handler kernel modules.

1.4. DM MULTIPATH COMPONENTS

Table 1.1, “DM Multipath Components” . describes the components of DM Multipath.

Table 1.1. DM Multipath Components

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 multipath devices. Normally started with **/etc/rc.sysinit**, it can also be started by a**udev** program whenever a block device is added.
multipathd daemon	Monitors paths; as paths fail and come back, it may initiate path group switches. Allows interactive changes to multipath devices. The daemon must be restarted following any changes to the **/etc/multipath.conf** file.
kpartx command	Creates device mapper devices for the partitions on a device. It is necessary to use this command for DOS-based partitions with DM Multipath. The **kpartx** command is provided in its own package, but the **device-mapper-multipath** package depends on it.

1.5. DM MULTIPATH SETUP OVERVIEW

DM Multipath includes compiled-in default settings that are suitable for common multipath configurations. The basic procedure for configuring your system with DM Multipath is as follows:

1. Install the **device-mapper-multipath** rpm.
2. Create the configuration file and enable multipathing with the **mpathconf** command. You can also start the multipath daemon with this command if you do not need to edit the configuration file.
3. If necessary, edit the **multipath.conf** configuration file to modify default values and save the updated file.
4. Start the multipath daemon.

For detailed setup instructions for multipath configuration see Chapter 3, *Setting Up DM Multipath*.

CHAPTER 2. MULTIPATH DEVICES

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

2.1. MULTIPATH DEVICE IDENTIFIERS

Each multipath device has a World Wide Identifier (WWID), which is guaranteed to be globally unique and unchanging. By default, the name of a multipath device is set to its WWID. Alternately, you can set the **user_friendly_names** option in the multipath configuration file, which sets the alias to a node-unique name of the form **mpath*n***.

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. When the **user_friendly_names** configuration option is set to **yes**, the name of the multipath device is set to **mpath*n***.

When new devices are brought under the control of DM Multipath, the new devices may be seen in two different places under the **/dev** directory: **/dev/mapper/mpath*n*** and **/dev/dm-*n***.

- The devices in **/dev/mapper** are created early in the boot process. Use these devices to access the multipathed devices, for example when creating logical volumes.
- Any devices of the form **/dev/dm-*n*** are for internal use only should never be used by the administrator directly.

For information on the multipath configuration defaults, including the **user_friendly_names** configuration option, see Section 4.3, “Configuration File Defaults”.

You can also set the name of a multipath device to a name of your choosing by using the **alias** option in the **multipaths** section of the multipath configuration file. For information on the **multipaths** section of the multipath configuration file, see Section 4.4, “Multipaths Device Configuration Attributes”.

2.2. CONSISTENT MULTIPATH DEVICE NAMES IN A CLUSTER

When the **user_friendly_names** configuration option is set to **yes**, the name of the multipath device is unique to a node, but it is not guaranteed to be the same on all nodes using the multipath device. Similarly, if you set the **alias** option for a device in the **multipaths** section of the **multipath.conf** configuration file, the name is not automatically consistent across all nodes in the cluster. This should not cause any difficulties if you use LVM to create logical devices from the multipath device, but if you require that your multipath device names be consistent in every node it is recommended that you not set the **user_friendly_names** option to **yes** and that you not configure aliases for the devices. By default, if you do not set **user_friendly_names** to **yes** or configure an alias for a device, a device name will be the WWID for the device, which is always the same.

If you want the system-defined user-friendly names to be consistent across all nodes in the cluster, however, you can follow this procedure:

1. Set up all of the multipath devices on one machine.
2. Disable all multipath devices on other machines by running the following commands:

```
# systemctl stop multipathd.service
# multipath -F
```

3. Copy the **/etc/multipath/bindings** file from the first machine to all the other machines in the cluster.

4. Re-enable the **multipathd** daemon on all the other machines in the cluster by running the following command:

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

If you add a new device, you will need to repeat this process.

Similarly, if you configure an alias for a device that you would like to be consistent across the nodes in the cluster, you should ensure that the **/etc/multipath.conf** file is the same for each node in the cluster by following the same procedure:

1. Configure the aliases for the multipath devices in the **multipath.conf** file on one machine.

2. Disable all multipath devices on other machines by running the following commands:

```
# systemctl stop multipathd.service
# multipath -F
```

3. Copy the **/etc/multipath.conf** file from the first machine to all the other machines in the cluster.

4. Re-enable the **multipathd** daemon on all the other machines in the cluster by running the following command:

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

When you add a new device you will need to repeat this process.

2.3. MULTIPATH DEVICE ATTRIBUTES

In addition to the **user_friendly_names** and **alias** options, a multipath device has numerous attributes. You can modify these attributes for a specific multipath device by creating an entry for that device in the **multipaths** section of the multipath configuration file. For information on the **multipaths** section of the multipath configuration file, see Section 4.4, “Multipaths Device Configuration Attributes” .

2.4. MULTIPATH DEVICES IN LOGICAL VOLUMES

After creating multipath devices, you can use the multipath device names just as you would use a physical device name when creating an LVM physical volume. For example, if **/dev/mapper/mpatha** is the name of a multipath device, the following command will mark **/dev/mapper/mpatha** as a physical volume.

```
pvcreate /dev/mapper/mpatha
```

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.

NOTE

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

When you create an LVM logical volume that uses active/passive multipath arrays as the underlying physical devices, you should include filters in the **/etc/lvm/lvm.conf** file to exclude the disks that underlie 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. For active/passive arrays that require a command to make the passive path active, LVM prints a warning message when this occurs.

To filter all SCSI devices in the LVM configuration file (**lvm.conf**), include the following filter in the **devices** section of the file.

```
filter = [ "r/block/", "r/disk/", "r/sd.*/", "a/.*/" ]
```

CHAPTER 3. SETTING UP DM MULTIPATH

This chapter provides step-by-step example procedures for configuring DM Multipath. It includes the following procedures:

- Basic DM Multipath setup
- Ignoring local disks
- Adding more devices to the configuration file
- Starting multipath in the **initramfs** file system

3.1. SETTING UP DM MULTIPATH

Before setting up DM Multipath on your system, ensure that your system has been updated and includes the **device-mapper-multipath** package.

You set up multipath with the **mpathconf** utility, which creates the multipath configuration file **/etc/multipath.conf**.

- 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 use the **/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf** file as the starting file.
- If the **/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf** file does not exist the **mpathconf** utility will create the **/etc/multipath.conf** file from scratch.

For more information on the **mpathconf** utility, see the **mpathconf(8)** man page.

If you do not need to edit the **/etc/multipath.conf** file, you can set up DM Multipath for a basic failover configuration by running the following command. This command enables the multipath configuration file and starts the **multipathd** daemon.

```
# mpathconf --enable --with_multipathd y
```

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

1. Enter the **mpathconf** command with the **--enable** option specified:

   ```
   # mpathconf --enable
   ```

   For information on additional options to the **mpathconf** command you may require, see the **mpathconf** man page or enter the **mpathconf** command with the **--help** option specified.

   ```
   # mpathconf --help
   usage: /sbin/mpathconf <command>

   Commands:
   Enable: --enable
   Disable: --disable
   Set user_friendly_names (Default y): --user_friendly_names <y|n>
   ```

```
Set find_multipaths (Default y): --find_multipaths <y|n>
Load the dm-multipath modules on enable (Default y): --with_module  <y|n>
start/stop/reload multipathd (Default n): --with_multipathd  <y|n>
```

2. Edit the **/etc/multipath.conf** file if necessary. The default settings for DM Multipath are compiled in to 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. For information on changing the values in the configuration file to something other than the defaults, see Chapter 4, *The DM Multipath Configuration File*.

   The initial defaults section of the configuration file configures your system so that the names of the multipath devices are of the form **mpath***n*; without this setting, the names of the multipath devices would be aliased to the WWID of the device.

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

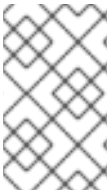
4. Execute the following command:

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

Since the value of **user_friendly_names** is set to **yes** in the configuration file, the multipath devices will be created as **/dev/mapper/mpath***n*. For information on setting the name of the device to an alias of your choosing, see Chapter 4, *The DM Multipath Configuration File* .

If you do not want to use user friendly names, you can enter the following command:

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



NOTE

If you find that 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.

3.2. IGNORING LOCAL DISKS WHEN GENERATING MULTIPATH DEVICES

Some machines have local SCSI cards for their internal disks. DM Multipath is not recommended for these devices. If you set the **find_multipaths** configuration parameter to **yes**, you should not have to blacklist these devices. For information on the **find_multipaths** configuration parameter, see Section 4.3, "Configuration File Defaults" .

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

1. Determine which disks are the internal disks and mark them as the ones to blacklist.

   In this example, **/dev/sda** is the internal disk. Note that as originally configured in the default multipath configuration file, executing the **multipath -v2** command shows the local disk, **/dev/sda**, in the multipath map.

This examples specifies the **-d** option of the **multipath** command to indicate that this is a dry run that will not create the multipath devices. For further information on the **multipath** command output, see Section 5.8, "Multipath Command Output".

```
# 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  [---------

: 3600a0b80001327d80000006d43621677 undef WINSYS,SF2372
size=12G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 2:0:0:0 sdb 8:16  undef ready  running
  `- 3:0:0:0 sdf 8:80 undef ready  running

: 3600a0b80001327510000009a436215ec undef WINSYS,SF2372
size=12G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 2:0:0:1 sdc 8:32 undef ready  running
  `- 3:0:0:1 sdg 8:96 undef ready  running

: 3600a0b80001327d800000070436216b3 undef WINSYS,SF2372
size=12G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 2:0:0:2 sdd 8:48 undef ready  running
  `- 3:0:0:2 sdg 8:112 undef ready  running

: 3600a0b80001327510000009b4362163e undef WINSYS,SF2372
size=12G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 2:0:0:3 sdd 8:64 undef ready  running
  `- 3:0:0:3 sdg 8:128 undef ready  running
```

2. In order to prevent the device mapper from mapping **/dev/sda** in its multipath maps, edit the blacklist section of the **/etc/multipath.conf** file to include this device. Although you could blacklist the **sda** device using a **devnode** type, that would not be a safe procedure since **/dev/sda** is not guaranteed to be the same on reboot. To blacklist individual devices, you can blacklist using the WWID of that device.

   Note that in the output to the **multipath -v2** command, the WWID of the **/dev/sda** device is SIBM-ESXSST336732LC___F3ET0EP0Q000072428BX1. To blacklist this device, include the following in the **/etc/multipath.conf** file.

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

3. After you have updated the **/etc/multipath.conf** file, you must manually tell the **multipathd** daemon to reload the file. The following command reloads the updated **/etc/multipath.conf** file.

```
# systemctl reload multipathd.service
```

3.3. CONFIGURING STORAGE DEVICES

By default, DM Multipath includes support for the most common storage arrays that themselves support DM Multipath. For information on the default configuration value, including supported devices, run either of the following commands.

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

If you need to add a 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.

For example, to add information about the HP Open-V series the entry looks like this. This example 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
        }
}
```

For more information on the **devices** section of the configuration file, see Section 4.5, “Configuration File Devices”.

3.4. SETTING UP MULTIPATHING IN THE INITRAMFS FILE SYSTEM

You can set up multipathing in the **initramfs** file system. After configuring multipath, you can rebuild the **initramfs** file system with the multipath configuration files by executing the **dracut** command with the following options:

```
# dracut --force --add multipath --include /etc/multipath
```

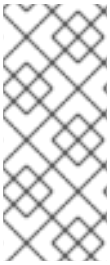
If you run multipath from the **initramfs** file system and you make any changes to the multipath configuration files, you must rebuild the **initramfs** file system for the changes to take effect.

CHAPTER 4. 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. For information on the default configuration values, including supported devices, run either of the following commands.

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

You can override the default configuration values for DM Multipath by editing the **/etc/multipath.conf** configuration file. If necessary, you can also add a storage array that is not supported by default to the configuration file.



NOTE

You can run set up multipathing in the **initramfs** file system. 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. For information on rebuilding the **initramfs** file system with multipath, see Section 3.4, "Setting Up Multipathing in the initramfs File System".

This chapter provides information on parsing and modifying the **multipath.conf** file. It contains sections on the following topics:

- Configuration file overview
- Configuration file blacklist
- Configuration file defaults
- Configuration file multipaths
- Configuration file devices

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.

Further information about the configuration file can be found on the **multipath.conf**(5) man page.

4.1. 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 candidates that would otherwise be blacklisted 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 **defaults** and **devices** 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.

When the system determines the attributes of a multipath device, first it checks the multipath settings, then the devices settings, then the multipath system defaults.

4.2. CONFIGURATION FILE BLACKLIST

The **blacklist** section of the multipath configuration file specifies the devices that will not be used when the system configures multipath devices. Devices that are blacklisted will not be grouped into a multipath device.

In older releases of Red Hat Enterprise Linux, multipath always tried to create a multipath device for every path that was not explicitly blacklisted. As of Red Hat Enterprise Linux 6, however, if the **find_multipaths** configuration parameter is set to **yes**, then multipath will create a device only if one of three conditions are met:

- There are at least two paths that are not blacklisted with the same WWID.
- The user manually forces 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 will automatically create 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 have to edit the multipath blacklist.

  If you have previously created a multipath device without using the **find_multipaths** parameter and then you later set the parameter to **yes**, you may 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 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/
```

With the **find_multipaths** parameter set to **yes**, you need to blacklist only the devices with multiple paths that you do not want to be multipathed. Because of this, it will generally not be necessary to blacklist devices.

If you do need to blacklist devices, you can do so according to the following criteria:

- By WWID, as described in Section 4.2.1, "Blacklisting by WWID"
- By device name, as described in Section 4.2.2, "Blacklisting By Device Name"
- By device type, as described in Section 4.2.3, "Blacklisting By Device Type"
- By **udev** property, as described in Section 4.2.4, "Blacklisting By udev Property (Red Hat Enterprise Linux 7.5 and Later)"
- By device protocol, as described in Section 4.2.5, "Blacklisting By Device Protocol (Red Hat Enterprise Linux 7.6 and Later)"

By default, a variety of device types are blacklisted, even after you comment out the initial blacklist section of the configuration file. For information, see Section 4.2.2, "Blacklisting By Device Name" .

4.2.1. Blacklisting by WWID

You can specify individual devices to blacklist by their World-Wide IDentification with a **wwid** entry in the **blacklist** section of the configuration file.

The following example shows the lines in the configuration file that would blacklist a device with a WWID of 26353900f02796769.

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

4.2.2. Blacklisting By Device Name

You can blacklist device types by device name so that they will not be grouped into a multipath device by specifying a **devnode** entry in the **blacklist** section of the configuration file.

The following example shows the lines in the configuration file that would blacklist all SCSI devices, since it blacklists all sd* devices.

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

You can use a **devnode** entry in the **blacklist** section of the configuration file to specify individual devices to blacklist rather than all devices of a specific type. This is not recommended, however, since 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, the following **devnode** entries are compiled in the default blacklist; the devices that these

entries blacklist do not generally support DM Multipath. To enable multipathing on any of these devices, you would need to specify them in the **blacklist_exceptions** section of the configuration file, as described in Section 4.2.6, "Blacklist Exceptions".

```
blacklist {
        devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
        devnode "^(td|ha)d[a-z]"
}
```

4.2.3. Blacklisting By Device Type

You can specify specific device types in the **blacklist** section of the configuration file with a **device** section. The following example blacklists all IBM DS4200 and HP devices.

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

4.2.4. Blacklisting By udev Property (Red Hat Enterprise Linux 7.5 and Later)

The **blacklist** and **blacklist_exceptions** sections of the **multipath.conf** configuration file support the **property** parameter. This parameter allows users to blacklist certain types of devices. The **property** parameter takes a regular expression string that is matched against the **udev** environment variable name for the device.

The following example blacklists all devices with the **udev** property **ID_ATA**.

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

4.2.5. Blacklisting By Device Protocol (Red Hat Enterprise Linux 7.6 and Later)

You can specify the protocol for a device to be excluded from multipathing in the **blacklist** section of the configuration file with a **protocol** section. 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 the command **multipathd show paths format "%d %P"**.

The following example blacklists all devices with an undefined protocol or an unknown SCSI transport type.

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

4.2.6. Blacklist Exceptions

You can use the **blacklist_exceptions** section of the configuration file to enable multipathing on devices that have been blacklisted by default.

For example, if you have a large number of devices and want to multipath only one of them (with the WWID of 3600d0230000000000e13955cc3757803), instead of individually blacklisting each of the devices except the one you want, you could instead blacklist all of them, and then allow only the one you want by adding the following lines to the **/etc/multipath.conf** file.

```
blacklist {
        wwid "*"
}

blacklist_exceptions {
        wwid "3600d0230000000000e13955cc3757803"
}
```

When specifying devices in the **blacklist_exceptions** section of the configuration file, you must specify the exceptions in the same way they were specified in the blacklist. For example, a WWID exception will not apply to devices specified by a **devnode** blacklist entry, even if the blacklisted device is associated with that WWID. Similarly, **devnode** exceptions apply only to **devnode** entries, and **device** exceptions apply only to device entries.

The **property** parameter works differently than the other **blacklist_exception** parameters. If the parameter is set, the device must have a **udev** variable that matches. Otherwise, the device is blacklisted. This parameter allows users to blacklist SCSI devices that multipath should ignore, such as USB sticks and local hard drives. To allow only 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)"
}
```

4.3. 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 configuration file includes a template of configuration defaults. This section is commented out, as follows.

```
#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
#}
```

To 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 so that it is **multibus** rather than 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
}
```

Table 4.1, "Multipath Configuration Defaults" describes the attributes that are set in the **defaults** section of the **multipath.conf** configuration file. These values are used by DM Multipath unless they are overwritten by the attributes specified in the **devices** and **multipaths** sections of the **multipath.conf** file.

Table 4.1. Multipath Configuration Defaults

Attribute	Description
polling_interval	Specifies the interval between two path checks in seconds. For properly functioning paths, the interval between checks will gradually increase to (4 * **polling_interval**). The default value is 5.
multipath_dir	The directory where the dynamic shared objects are stored. The default value is system dependent, commonly **/lib/multipath**.

Attribute	Description
find_multipaths	Defines the mode for setting up multipath devices. If this parameter is set to **yes**, then multipath will not try to create a device for every path that is not blacklisted. Instead multipath will create a device only if one of three conditions are met: - There are at least two paths that are not blacklisted with the same WWID. - The user manually forces 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. Whenever a multipath device is created with **find_multipaths** set, multipath remembers the WWID of the device so that it will automatically create 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 edit the multipath blacklist. For instructions on the procedure to follow if you have previously created multipath devices when the **find_multipaths** parameter was not set, see Section 4.2, "Configuration File Blacklist". The default value is **no**. The default **multipath.conf** file created by **mpathconf**, however, will enable **find_multipaths** as of Red Hat Enterprise Linux 7.
reassign_maps	Enable reassigning of device-mapper maps. With this option, The **multipathd** daemon will remap existing device-mapper maps to always point to the multipath device, not the underlying block devices. Possible values are **yes** and **no**. The default value is **yes**.
verbosity	The default verbosity. Higher values increase the verbosity level. Valid levels are between 0 and 6. The default value is **2**.

Attribute	Description
path_selector	Specifies the default algorithm to use in determining what path to use for the next I/O operation. Possible values include: **round-robin 0**: Loop through every path in the path group, sending the same amount of I/O to each. **queue-length 0**: Send the next bunch of I/O down the path with the least number of outstanding I/O requests. **service-time 0**: Send the next bunch of I/O down the path with the shortest estimated service time, which is determined by dividing the total size of the outstanding I/O to each path by its relative throughput. The default value is **service-time 0**.
path_grouping_policy	Specifies the default path grouping policy to apply to unspecified multipaths. Possible values include: **failover**: 1 path per priority group. **multibus**: all valid paths in 1 priority group. **group_by_serial**: 1 priority group per detected serial number. **group_by_prio**: 1 priority group per path priority value. Priorities are determined by callout programs specified as global, per-controller, or per-multipath options. **group_by_node_name**: 1 priority group per target node name. Target node names are fetched in **/sys/class/fc_transport/target*/node_name**. The default value is **failover**.

Attribute	Description
prio	Specifies the default function to call to obtain a path priority value. For example, the ALUA bits in SPC-3 provide an exploitable **prio** value. Possible values include: **const**: Set a priority of 1 to all paths. **emc**: Generate the path priority for EMC arrays. **alua**: Generate the path priority based on the SCSI-3 ALUA settings. As of Red Hat Enterprise Linux 7.3, if you specify **prio "alua exclusive_pref_bit"** in your device configuration, multipath will create a path group that contains only the path with the **pref** bit set and will give that path group the highest priority. **ontap**: Generate the path priority for NetApp arrays. **rdac**: Generate the path priority for LSI/Engenio RDAC controller. **hp_sw**: Generate the path priority for Compaq/HP controller in active/standby mode. **hds**: Generate the path priority for Hitachi HDS Modular storage arrays. The default value is **const**.

Attribute	Description
features	The default extra features of multipath devices, using the format: "*number_of_features_plus_arguments feature1 ...*". Possible values for **features** include: **queue_if_no_path**, which is the same as setting **no_path_retry** to **queue**. For information on issues that may arise when using this feature, see Section 5.7, "Issues with queue_if_no_path feature". **retain_attached_hw_handler**: If this parameter is set to **yes** and the SCSI layer has already attached a hardware handler to the path device, multipath will not force the device to use the **hardware_handler** specified by the **multipath.conf** file. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler as usual. The default value is **no**. **pg_init_retries *n***: Retry path group initialization up to *n* times before failing where 1 <= *n* <= 50. **pg_init_delay_msecs *n***: Wait *n* milliseconds between path group initialization retries where 0 <= *n* <= 60000.
path_checker	Specifies the default method used to determine the state of the paths. Possible values include: **readsector0**: Read the first sector of the device. **tur**: Issue a **TEST UNIT READY** command to the device. **emc_clariion**: Query the EMC Clariion specific EVPD page 0xC0 to determine the path. **hp_sw**: Check the path state for HP storage arrays with Active/Standby firmware. **rdac**: Check the path state for LSI/Engenio RDAC storage controller. **directio**: Read the first sector with direct I/O. The default value is **directio**.

Attribute	Description
failback	Manages path group failback. A value of **immediate** specifies immediate failback to the highest priority path group that contains active paths. A value of **manual** specifies that there should not be immediate failback but that failback can happen only with operator intervention. A value of **followover** specifies that automatic failback should be performed when the first path of a path group becomes active. This keeps a node from automatically failing back when another node requested the failover. A numeric value greater than zero specifies deferred failback, expressed in seconds. The default value is **manual**.
rr_min_io	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group. This setting is only for systems running kernels older than 2.6.31. Newer systems should use **rr_min_io_rq**. The default value is 1000.
rr_min_io_rq	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group, using request-based device-mapper-multipath. This setting should be used on systems running current kernels. On systems running kernels older than 2.6.31, use **rr_min_io**. The default value is 1.
rr_weight	If set to **priorities**, then instead of sending**rr_min_io** requests to a path before calling **path_selector** to choose the next path, the number of requests to send is determined by **rr_min_io** times the path's priority, as determined by the **prio** function. If set to **uniform**, all path weights are equal. The default value is **uniform**.
no_path_retry	A numeric value for this attribute specifies the number of times the system should attempt to use a failed path before disabling queuing. A value of **fail** indicates immediate failure, without queuing. A value of **queue** indicates that queuing should not stop until the path is fixed. The default value is 0.

Attribute	Description
user_friendly_names	If set to **yes**, specifies that the system should use the **/etc/multipath/bindings** file to assign a persistent and unique alias to the multipath, in the form of **mpath*n***. If set to **no**, specifies that the system should use the WWID as the alias for the multipath. In either case, what is specified here will be overridden by any device-specific aliases you specify in the **multipaths** section of the configuration file. The default value is **no**.
queue_without_daemon	If set to **no**, the **multipathd** daemon will disable queuing for all devices when it is shut down. The default value is **no**.
flush_on_last_del	If set to **yes**, the **multipathd** daemon will disable queuing when the last path to a device has been deleted. The default value is **no**.
max_fds	Sets the maximum number of open file descriptors that can be opened by multipath and the **multipathd** daemon. This is equivalent to the **ulimit -n** command. As of the Red Hat Enterprise Linux 6.3 release, the default value is **max**, which sets this to the system limit from **/proc/sys/fs/nr_open**. For earlier releases, if this is not set the maximum number of open file descriptors is taken from the calling process; it is usually 1024. To be safe, this should be set to the maximum number of paths plus 32, if that number is greater than 1024.
checker_timeout	The timeout to use for prioritizers and path checkers that issue SCSI commands with an explicit timeout, in seconds. The default value is taken from **sys/block/sd*x*/device/timeout**.
fast_io_fail_tmo	The number of seconds the SCSI layer will wait after a problem has been detected on an FC remote port before failing I/O to devices on that remote port. This value should be smaller than the value of **dev_loss_tmo**. Setting this to **off** will disable the timeout. The default value is determined by the OS.
dev_loss_tmo	The number of seconds the SCSI layer will wait after a problem has been detected on an FC remote port before removing it from the system. Setting this to infinity will set this to 2147483647 seconds, or 68 years. The default value is determined by the OS.

Attribute	Description
hw_string_match	Each device configuration in the **devices** section of the **multipath.conf** file will either create its own device configuration or it will modify one of the built-in device configurations. If **hw_string_match** is set to **yes**, then if the vendor, product, and revision strings in a user's device configuration exactly match those strings in a built-in device configuration, the built-in configuration is modified by the options in the user's configuration. Otherwise, the user's device configuration is treated as a new configuration. If **hw_string_match** is set to **no**, a regular expression match is used instead of a string match. The **hw_string_match** parameter is set to **no** by default.
retain_attached_hw_handler	If this parameter is set to **yes** and the SCSI layer has already attached a hardware handler to the path device, multipath will not force the device to use the **hardware_handler** specified by the **multipath.conf** file. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler as usual. The default value is **no**.
detect_prio	If this is set to **yes**, multipath will first check if the device supports ALUA, and if so it will automatically assign the device the **alua** prioritizer. If the device does not support ALUA, it will determine the prioritizer as it always does. The default value is **no**.
uid_attribute	Provides a unique path identifier. The default value is **ID_SERIAL**.
force_sync	(Red Hat Enterprise Linux Release 7.1 and later) If this is set to "yes", it prevents path checkers from running in async mode. This means that only one checker will run at a time. This is useful in the case where many **multipathd** checkers running in parallel causes significant CPU pressure. The default value is **no**.
delay_watch_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, the **multipathd** daemon will watch paths that have recently become valid for the specified number of checks. If they fail again while they are being watched, when they next become valid they will not be used until they have stayed up for the number of consecutive checks specified with **delay_wait_checks**. This allows you to keep paths that may be unreliable from immediately being put back into use as soon as they come back online. The default value is **no**.
delay_wait_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, when a device that has recently come back online fails again within the number of checks specified with **delay_watch_checks**, the next time it comes back online it will be marked and delayed and it will not be used until it has passed the number of checks specified in **delay_wait_checks**. The default value is **no**.

Attribute	Description
ignore_new_boot_devs	(Red Hat Enterprise Linux Release 7.2 and later) If set to **yes**, when the node is still in the **initramfs** file system during early boot, multipath will not create any devices whose WWIDs do not already exist in the **initramfs copy** of the **/etc/multipath/wwids**. This feature can be used for booting up during installation, when multipath would otherwise attempt to set itself up on devices that it did not claim when they first appeared by means of the **udev** rules. This parameter can be set to **yes** or **no**. If unset, it defaults to **no**.
retrigger_tries, **retrigger_delay**	(Red Hat Enterprise Linux Release 7.2 and later) The **retrigger_tries** and **retrigger_delay** parameters are used in conjunction to make **multipathd** retrigger uevents if **udev** failed to completely process the original ones, leaving the device unusable by multipath. The **retrigger_tries** parameter sets the number of times that multipath will try to retrigger a **uevent** if a device has not been completely set up. The **retrigger_delay** parameter sets the number of seconds between retries. Both of these options accept numbers greater than or equal to zero. Setting the **retrigger_tries** parameter to zero disables retries. Setting the **retrigger_delay** parameter to zero causes the **uevent** to be reissued on the next loop of the path checker. If the **retrigger_tries** parameter is unset, it defaults to 3. If the **retrigger_delay** parameter is unset, it defaults to 10.
new_bindings_in_boot	(Red Hat Enterprise Linux Release 7.2 and later) The **new_bindings_in_boot** parameter is used to keep multipath from giving out a **user_friendly_name** in the **initramfs** file system that was already given out by the bindings file in the regular file system, an issue that can arise since the **user_friendly_names** bindings in the **initramfs** file system get synced with the bindings in the regular file system only when the **initramfs** file system is remade. When this parameter is set to **no** multipath will not create any new bindings in the **initramfs** file system. If a device does not already have a binding in the **initramfs** copy of **/etc/multipath/bindings**, multipath will use its WWID as an alias instead of giving it a **user_friendly_name**. Later in boot, after the node has mounted the regular filesystem, multipath will give out a **user_friendly_name** to the device. This parameter can be set to **yes** or **no**. If unset, it defaults to **no**.
config_dir	(Red Hat Enterprise Linux Release 7.2 and later) If set to anything other than **""**, multipath will search this directory alphabetically for files ending in ".conf" and it will read configuration information from them, just as if the information were in the **/etc/multipath.conf** file. This allows you to have one main configuration that you share between machines in addition to a separate machine-specific configuration file or files. The **config_dir** parameter must either be **""** or a fully qualified directory name. This parameter can be set only in the main **/etc/multipath.conf** file and not in one of the files specified in the **config_dir** file itself. The default value is **/etc/multipath/conf.d**.

Attribute	Description
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 ensures that if a multipathed device is in use when a regular remove is performed and the remove fails, the device will automatically be removed when the last user closes the device. The default value is **no**.
log_checker_err	If set to **once**, multipathd logs the first path checker error at verbosity level 2. Any later errors are logged at verbosity level 3 until the device is restored. If it is set to **always**, multipathd always logs the path checker error at verbosity level 2. The default value is **always**.
skip_kpartx	(Red Hat Enterprise Linux Release 7.3 and later) If set to **yes**, **kpartx** will not automatically create partitions on the device. This allows users to create a multipath device without creating partitions, even if the device has a partition table. The default value of this option is **no**.
max_sectors_kb	(Red Hat Enterprise Linux Release 7.4 and later) Sets the **max_sectors_kb** device queue parameter to the specified value on all underlying paths of a multipath device before the multipath device is first activated. When a multipath device is created, the device inherits the **max_sectors_kb** value from the path devices. Manually raising this value for the multipath device or lowering this value for the path devices can cause multipath to create I/O operations larger than the path devices allow. Using the **max_sectors_kb** parameter is an easy way to set these values before a multipath device is created on top of the path devices and prevent invalid-sized I/O operations from being passed. If this parameter is not set by the user, the path devices have it set by their device driver, and the multipath device inherits it from the path devices.
remove_retries	(Red Hat Enterprise Linux Release 7.4 and later) Sets how may times multipath will retry removing a device that is in use. Between each attempt, multipath will sleep 1 second. The default value is 0, which means that multipath will not retry the remove.
disable_changed_wwids	(Red Hat Enterprise Linux Release 7.4 and later) If set to **yes** and the WWID of a path device changes while it is part of a multipath device, multipath will disable access to the path device until the WWID of the path is restored to the WWID of the multipath device. The default value is **no**, which does not check if a path's WWID has changed.
detect_path_checker	(Red Hat Enterprise Linux Release 7.4 and later) If set to **yes**, multipath will try to detect if the device supports ALUA. If so, the device will automatically use the **tur** path checker. If not, the path_checker will be selected as usual. The default value is **no**.

Attribute	Description
reservation_key	This is the service action reservation key used by **mpathpersist**. It must be set for all multipath devices using persistent reservations, and it must be the same as the **RESERVATION KEY** field of the **PERSISTENT RESERVE OUT** parameter list which contains an 8-byte value provided by the application client to the device server to identify the **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. As of Red Hat Enterprise Linux Release 7.5, this parameter can be set to **file**, which will store the **RESERVATION KEY** registered by **mpath-persist** in the **prkeys** file. The **multipathd** daemon will then use this key to register additional paths as they appear. When the registration is removed, the **RESERVATION KEY** is removed from the **prkeys** file. It is unset by default.
prkeys_file	(Red Hat Enterprise Linux Release 7.5 and later) The full path name of the **prkeys** file, which is used by the**multipathd** daemon to keep track of the reservation key used for a specific WWID when the **reservation_key** parameter is set to **file**. The default value is **/etc/multipath/prkeys**.
all_tg_pt	(Red Hat Enterprise Linux Release 7.6 and later) If this option is set to **yes**, when **mpathpersist** registers keys it will treat a key registered from one host to one target port as going from one host to all target ports. This must be set to **yes** to successfully use **mpathpersist** on arrays that automatically set and clear registration keys on all target ports from a host, instead of per target port per host. The default value is **no**.

4.4. MULTIPATHS DEVICE CONFIGURATION ATTRIBUTES

Table 4.2, “Multipath Attributes” shows the attributes that you can set in the **multipaths** section of the **multipath.conf** configuration file for each specific multipath device. These attributes apply only to the one specified multipath. These defaults are used by DM Multipath and override attributes set in the **defaults** and **devices** sections of the **multipath.conf** file.

Table 4.2. Multipath Attributes

Attribute	Description
wwid	Specifies the WWID of the multipath device to which the **multipath** attributes apply. This parameter is mandatory for this section of the **multipath.conf** file.

Attribute	Description
alias	Specifies the symbolic name for the multipath device to which the **multipath** attributes apply. If you are using **user_friendly_names**, do not set this value to **mpath*n***; this may conflict with an automatically assigned user friendly name and give you incorrect device node names.
path_grouping_policy	Specifies the default path grouping policy to apply to unspecified multipaths. Possible values include: **failover** = 1 path per priority group **multibus** = all valid paths in 1 priority group **group_by_serial** = 1 priority group per detected serial number **group_by_prio** = 1 priority group per path priority value **group_by_node_name** = 1 priority group per target node name
path_selector	Specifies the default algorithm to use in determining what path to use for the next I/O operation. Possible values include: **round-robin 0**: Loop through every path in the path group, sending the same amount of I/O to each. **queue-length 0**: Send the next bunch of I/O down the path with the least number of outstanding I/O requests. **service-time 0**: Send the next bunch of I/O down the path with the shortest estimated service time, which is determined by dividing the total size of the outstanding I/O to each path by its relative throughput.

Attribute	Description
failback	Manages path group failback. A value of **immediate** specifies immediate failback to the highest priority path group that contains active paths. A value of **manual** specifies that there should not be immediate failback but that failback can happen only with operator intervention. A value of **followover** specifies that automatic failback should be performed when the first path of a path group becomes active. This keeps a node from automatically failing back when another node requested the failover. A numeric value greater than zero specifies deferred failback, expressed in seconds.
prio	Specifies the default function to call to obtain a path priority value. For example, the ALUA bits in SPC-3 provide an exploitable **prio** value. Possible values include: **const**: Set a priority of 1 to all paths. **emc**: Generate the path priority for EMC arrays. **alua**: Generate the path priority based on the SCSI-3 ALUA settings. As of Red Hat Enterprise Linux 7.3, if you specify **prio "alua exclusive_pref_bit"** in your device configuration, multipath will create a path group that contains only the path with the **pref** bit set and will give that path group the highest priority. **ontap**: Generate the path priority for NetApp arrays. **rdac**: Generate the path priority for LSI/Engenio RDAC controller. **hp_sw**: Generate the path priority for Compaq/HP controller in active/standby mode. **hds**: Generate the path priority for Hitachi HDS Modular storage arrays.

Attribute	Description
features	The default extra features of multipath devices, using the format: "*number_of_features_plus_arguments feature1 ...*". Possible values for **features** include: **queue_if_no_path**, which is the same as setting **no_path_retry** to **queue**. For information on issues that may arise when using this feature, see Section 5.7, "Issues with queue_if_no_path feature". **retain_attached_hw_handler**: If this parameter is set to **yes** and the SCSI layer has already attached a hardware handler to the path device, multipath will not force the device to use the **hardware_handler** specified by the **multipath.conf** file. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler as usual. The default value is **no**. **pg_init_retries *n***: Retry path group initialization up to *n* times before failing where 1 <= *n* <= 50. **pg_init_delay_msecs *n***: Wait *n* milliseconds between path group initialization retries where 0 <= *n* <= 60000.
no_path_retry	A numeric value for this attribute specifies the number of times the system should attempt to use a failed path before disabling queuing. A value of **fail** indicates immediate failure, without queuing. A value of **queue** indicates that queuing should not stop until the path is fixed.
rr_min_io	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group. This setting is only for systems running kernels older that 2.6.31. Newer systems should use **rr_min_io_rq**. The default value is 1000.
rr_min_io_rq	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group, using request-based device-mapper-multipath. This setting should be used on systems running current kernels. On systems running kernels older than 2.6.31, use **rr_min_io**. The default value is 1.
rr_weight	If set to **priorities**, then instead of sending **rr_min_io** requests to a path before calling **path_selector** to choose the next path, the number of requests to send is determined by **rr_min_io** times the path's priority, as determined by the **prio** function. If set to **uniform**, all path weights are equal.

Attribute	Description
flush_on_last_del	If set to **yes**, then multipath will disable queuing when the last path to a device has been deleted.
user_friendly_names	If set to **yes**, specifies that the system should use the **/etc/multipath/bindings** file to assign a persistent and unique alias to the multipath, in the form of **mpath*n***. If set to **no**, specifies that the system should use use the WWID as the alias for the multipath. In either case, what is specified here will be overridden by any device-specific aliases you specify in the **multipaths** section of the configuration file.
delay_watch_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, the **multipathd** daemon will watch paths that have recently become valid for the specified number of checks. If they fail again while they are being watched, when they next become valid they will not be used until they have stayed up for the number of consecutive checks specified with **delay_wait_checks**. This allows you to keep paths that may be unreliable from immediately being put back into use as soon as they come back online.
delay_wait_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, when a device that has recently come back online fails again within the number of checks specified with **delay_watch_checks**, the next time it comes back online it will be marked and delayed and it will not be used until it has passed the number of checks specified in **delay_wait_checks**.
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 ensures that if a multipathed device is in use when a regular remove is performed and the remove fails, the device will automatically be removed when the last user closes the device.
skip_kpartx	(Red Hat Enterprise Linux Release 7.3 and later) If set to **yes**, **kpartx** will not automatically create partitions on the device. This allows users to create a multipath device without creating partitions, even if the device has a partition table.
max_sectors_kb	(Red Hat Enterprise Linux Release 7.4 and later) Sets the **max_sectors_kb** device queue parameter to the specified value on all underlying paths of a multipath device before the multipath device is first activated. When a multipath device is created, the device inherits the **max_sectors_kb** value from the path devices. Manually raising this value for the multipath device or lowering this value for the path devices can cause multipath to create I/O operations larger than the path devices allow. Using the **max_sectors_kb** parameter is an easy way to set these values before a multipath device is created on top of the path devices and prevent invalid-sized I/O operations from being passed If this parameter is not set by the user, the path devices have it set by their device driver, and the multipath device inherits it from the path devices.

Attribute	Description
reservation_key	This is the service action reservation key used by **mpathpersist**. It must be set for all multipath devices using persistent reservations, and it must be the same as the **RESERVATION KEY** field of the **PERSISTENT RESERVE OUT** parameter list which contains an 8-byte value provided by the application client to the device server to identify the **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. As of Red Hat Enterprise Linux Release 7.5, this parameter can be set to **file**, which will store the **RESERVATION KEY** registered by **mpath-persist** in the **prkeys_file** file. The **multipathd** daemon will then use this key to register additional paths as they appear. When the registration is removed, the **RESERVATION KEY** is removed from the **prkeys_file** file.

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** attributes 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
        }
}
```

4.5. CONFIGURATION FILE DEVICES

Table 4.3, “Device Attributes” shows the attributes that you can set for each individual storage device in the **devices** section of the **multipath.conf** configuration file. These attributes are used by DM Multipath 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 **defaults** section of the **multipath.conf** file.

Many devices that support multipathing are included by default in a multipath configuration. For information on the default configuration value, including supported devices, run either of the following commands.

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

You probably will not need to modify the values for these devices, but if you do 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.

To add a device that is not configured automatically by default to this section of the configuration file, you need to set the **vendor** and **product** parameters. You can find these values by looking at **/sys/block/*device_name*/device/vendor** and **/sys/block/*device_name*/device/model** where *device_name* is the device to be multipathed, as in the following example:

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

The additional parameters to specify depend on your specific device. If the device is active/active, you will usually not need to set additional parameters. You may want to set **path_grouping_policy** to **multibus**. Other parameters you may need to set are **no_path_retry** and **rr_min_io**, as described in Table 4.3, “Device Attributes”.

If the device is active/passive, but 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 almost always means that you set the **path_checker** to **tur**; this 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 may not be able to configure the device for multipath.

Table 4.3. Device Attributes

Attribute	Description
vendor	Specifies the vendor name of the storage device to which the **device** attributes apply, for example **COMPAQ**.
product	Specifies the product name of the storage device to which the **device** attributes apply, for example **HSV110 (C)COMPAQ**.
revision	Specifies the product revision identifier of the storage device.
product_blacklist	Specifies a regular expression used to blacklist devices by product.
alias_prefix	The **user_friendly_names** prefix to use for this device type, instead of the default "mpath".

Attribute	Description
hardware_handler	Specifies a module that will be used to perform hardware specific actions when switching path groups or handling I/O errors. Possible values include: **1 emc**: hardware handler for EMC storage arrays. **1 alua**: hardware handler for SCSI-3 ALUA arrays. **1 hp_sw**: hardware handler for Compaq/HP controllers. **1 rdac**: hardware handler for the LSI/Engenio RDAC controllers.
path_grouping_policy	Specifies the default path grouping policy to apply to unspecified multipaths. Possible values include: **failover** = 1 path per priority group **multibus** = all valid paths in 1 priority group **group_by_serial** = 1 priority group per detected serial number **group_by_prio** = 1 priority group per path priority value **group_by_node_name** = 1 priority group per target node name
path_selector	Specifies the default algorithm to use in determining what path to use for the next I/O operation. Possible values include: **round-robin 0**: Loop through every path in the path group, sending the same amount of I/O to each. **queue-length 0**: Send the next bunch of I/O down the path with the least number of outstanding I/O requests. **service-time 0**: Send the next bunch of I/O down the path with the shortest estimated service time, which is determined by dividing the total size of the outstanding I/O to each path by its relative throughput.

Attribute	Description
path_checker	Specifies the default method used to determine the state of the paths. Possible values include: **readsector0**: Read the first sector of the device. **tur**: Issue a **TEST UNIT READY** to the device. **emc_clariion**: Query the EMC Clariion specific EVPD page 0xC0 to determine the path. **hp_sw**: Check the path state for HP storage arrays with Active/Standby firmware. **rdac**: Check the path stat for LSI/Engenio RDAC storage controller. **directio**: Read the first sector with direct I/O.
features	The default extra features of multipath devices, using the format: "*number_of_features_plus_arguments feature1 ...*". Possible values for **features** include: **queue_if_no_path**, which is the same as setting **no_path_retry** to **queue**. For information on issues that may arise when using this feature, see Section 5.7, "Issues with queue_if_no_path feature". **retain_attached_hw_handler**: If this parameter is set to **yes** and the SCSI layer has already attached a hardware handler to the path device, multipath will not force the device to use the **hardware_handler** specified by the **multipath.conf** file. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler as usual. **pg_init_retries *n***: Retry path group initialization up to *n* times before failing where 1 <= *n* <= 50. **pg_init_delay_msecs *n***: Wait *n* milliseconds between path group initialization retries where 0 <= *n* <= 60000.

Attribute	Description
prio	Specifies the default function to call to obtain a path priority value. For example, the ALUA bits in SPC-3 provide an exploitable **prio** value. Possible values include: **const**: Set a priority of 1 to all paths. **emc**: Generate the path priority for EMC arrays. **alua**: Generate the path priority based on the SCSI-3 ALUA settings. As of Red Hat Enterprise Linux 7.3, if you specify **prio "alua exclusive_pref_bit"** in your device configuration, multipath will create a path group that contains only the path with the **pref** bit set and will give that path group the highest priority. **ontap**: Generate the path priority for NetApp arrays. **rdac**: Generate the path priority for LSI/Engenio RDAC controller. **hp_sw**: Generate the path priority for Compaq/HP controller in active/standby mode. **hds**: Generate the path priority for Hitachi HDS Modular storage arrays.
failback	Manages path group failback. A value of **immediate** specifies immediate failback to the highest priority path group that contains active paths. A value of **manual** specifies that there should not be immediate failback but that failback can happen only with operator intervention. A value of **followover** specifies that automatic failback should be performed when the first path of a path group becomes active. This keeps a node from automatically failing back when another node requested the failover. A numeric value greater than zero specifies deferred failback, expressed in seconds.

Attribute	Description
rr_weight	If set to **priorities**, then instead of sending**rr_min_io** requests to a path before calling **path_selector** to choose the next path, the number of requests to send is determined by **rr_min_io** times the path's priority, as determined by the **prio** function. If set to**uniform**, all path weights are equal.
no_path_retry	A numeric value for this attribute specifies the number of times the system should attempt to use a failed path before disabling queuing. A value of **fail** indicates immediate failure, without queuing. A value of **queue** indicates that queuing should not stop until the path is fixed.
rr_min_io	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group. This setting is only for systems running kernels older that 2.6.31. Newer systems should use **rr_min_io_rq**. The default value is 1000.
rr_min_io_rq	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group, using request-based device-mapper-multipath. This setting should be used on systems running current kernels. On systems running kernels older than 2.6.31, use **rr_min_io**. The default value is 1.
fast_io_fail_tmo	The number of seconds the SCSI layer will wait after a problem has been detected on an FC remote port before failing I/O to devices on that remote port. This value should be smaller than the value of **dev_loss_tmo**. Setting this to **off** will disable the timeout.
dev_loss_tmo	The number of seconds the SCSI layer will wait after a problem has been detected on an FC remote port before removing it from the system. Setting this to infinity will set this to 2147483647 seconds,or 68 years.
flush_on_last_del	If set to **yes**, the **multipathd** daemon will disable queuing when the last path to a device has been deleted.
user_friendly_names	If set to **yes**, specifies that the system should use the **/etc/multipath/bindings** file to assign a persistent and unique alias to the multipath, in the form of **mpath*n***. If set to **no**, specifies that the system should use use the WWID as the alias for the multipath. In either case, what is specified here will be overridden by any device-specific aliases you specify in the **multipaths** section of the configuration file. The default value is **no**.

Attribute	Description
retain_attached_hw_handler	If this parameter is set to **yes** and the SCSI layer has already attached a hardware handler to the path device, multipath will not force the device to use the **hardware_handler** specified by the **multipath.conf** file. If the SCSI layer has not attached a hardware handler, multipath will continue to use its configured hardware handler as usual. The default value is **no**.
detect_prio	If this is set to **yes**, multipath will first check if the device supports ALUA, and if so it will automatically assign the device the **alua** prioritizer. If the device does not support ALUA, it will determine the prioritizer as it always does.
uid_attribute	Provides a unique path identifier.
delay_watch_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, the **multipathd** daemon will watch paths that have recently become valid for the specified number of checks. If they fail again while they are being watched, when they next become valid they will not be used until they have stayed up for the number of consecutive checks specified with **delay_wait_checks**. This allows you to keep paths that may be unreliable from immediately being put back into use as soon as they come back online.
delay_wait_checks	(Red Hat Enterprise Linux Release 7.2 and later) If set to a value greater than 0, when a device that has recently come back online fails again within the number of checks specified with **delay_watch_checks**, the next time it comes back online it will be marked and delayed and it will not be used until it has passed the number of checks specified in **delay_wait_checks**.
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 ensures that if a multipathed device is in use when a regular remove is performed and the remove fails, the device will automatically be removed when the last user closes the device.
skip_kpartx	(Red Hat Enterprise Linux Release 7.3 and later) If set to **yes**, **kpartx** will not automatically create partitions on the device. This allows users to create a multipath device without creating partitions, even if the device has a partition table.

Attribute	Description
max_sectors_kb	(Red Hat Enterprise Linux Release 7.4 and later) Sets the **max_sectors_kb** device queue parameter to the specified value on all underlying paths of a multipath device before the multipath device is first activated. When a multipath device is created, the device inherits the **max_sectors_kb** value from the path devices. Manually raising this value for the multipath device or lowering this value for the path devices can cause multipath to create I/O operations larger than the path devices allow. Using the **max_sectors_kb** parameter is an easy way to set these values before a multipath device is created on top of the path devices and prevent invalid-sized I/O operations from being passed If this parameter is not set by the user, the path devices have it set by their device driver, and the multipath device inherits it from the path devices.
detect_path_checker	(Red Hat Enterprise Linux Release 7.4 and later) If set to **yes**, multipath will try to detect if the device supports ALUA. If so, the device will automatically use the **tur** path checker. If not, the path_checker will be selected as usual.
all_devs	When this parameter is set to **yes**, all of the options set in this device configuration will override the values for those options in all of the other device configurations, both the ones in the configuration file and the built-in defaults.
all_tg_pt	(Red Hat Enterprise Linux Release 7.6 and later) If this option is set to **yes**, when **mpathpersist** registers keys it will treat a key registered from one host to one target port as going from one host to all target ports. This must be set to **yes** to successfully use **mpathpersist** on arrays that automatically set and clear registration keys on all target ports from a host, instead of per target port per host.

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
# }
#}
```

The following configuration sets **no_path_retry** to **fail** for all of the built-in device configurations.

```
devices {
 device {
  all_devs yes
  no_path_retry fail
 }
}
```

CHAPTER 5. DM MULTIPATH ADMINISTRATION AND TROUBLESHOOTING

This chapter provides information on administering DM Multipath on a running system.

5.1. AUTOMATIC CONFIGURATION FILE GENERATION WITH MULTIPATH HELPER

You can generate a basic configuration for multipath devices on Red Hat Enterprise Linux with the Multipath Helper application. The application gives you options to create multipath configurations with custom aliases, device blacklists, and settings for the characteristics of individual multipath devices. Upon completion, the application generates an installation script that includes the configuration parameters you selected and it provides a **multipath.conf** configuration file for review.

The Multipath Helper application can be found at https://access.redhat.com/labsinfo/multipathhelper.

5.2. RESIZING AN ONLINE MULTIPATH DEVICE

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

1. Resize your physical device.
2. Execute the following command to find the paths to the 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 by executing the **multipathd resize** command:

```
# multipathd resize map multipath_device
```

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

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

5.3. MOVING ROOT FILE SYSTEMS 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. This section documents the procedure for moving from a single-path to a multipathed device.

After ensuring that you have installed the **device-mapper-multipath** package, perform the following 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**:

   ```
   # mpathconf --enable
   ```

   For further information on using the **mpathconf** command to set up multipathing, see Section 3.1, "Setting Up DM Multipath".

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 Section 4.2, "Configuration File Blacklist".

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** will allow 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. To confirm that your configuration file is set up correctly, you can enter 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 if 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. To rebuild the **initramfs** file system with **multipath**, execute the **dracut** command with the following options:

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

6. Shut the machine down.

7. Configure the FC switch so that other paths are visible to the machine.

8. Boot the machine.

9. Check whether the root file system ('/') is on the multipathed device.

5.4. MOVING SWAP FILE SYSTEMS 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 may need to edit the **/etc/fstab** file to switch to the appropriate multipath device name.

1. Determine the WWID number of the swap device by running the **/sbin/multipath** command with the **-v3** option. The output from the command should show the swap device in the paths list.

   You should look in the command output for a line of the following format, showing the swap device:

   ```
   WWID  H:B:T:L devname MAJOR:MINOR
   ```

   For example, if your swap file system is set up on **sda** or one of its partitions, you would see a line in the output such as the following:

   ```
   ===== paths list =====
   ...
   1ATA     WDC WD800JD-75MSA3                      WD-WMAM9F 1:0:0:0 sda 8:0
   ...
   ```

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

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

3. 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/sda2 swap                  swap    defaults        0 0
   ```

   You would change the entry to the following:

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

5.5. THE MULTIPATH DAEMON

If you find you have trouble implementing a multipath configuration, you should ensure that the multipath daemon is running, as described in Chapter 3, *Setting Up DM Multipath*. The **multipathd** daemon must be running in order to use multipathed devices.

5.6. ISSUES WITH LARGE NUMBER OF LUNS

When a large number of LUNs are added to a node, using multipathed devices can significantly increase the time it takes for the **udev** device manager to create device nodes for them. If you experience this problem, you can correct it by deleting the following line in **/etc/udev/rules.d/40-multipath.rules**:

```
KERNEL!="dm-[0-9]*", ACTION=="add", PROGRAM=="/bin/bash -c '/sbin/lsmod | /bin/grep
^dm_multipath'", RUN+="/sbin/multipath -v0 %M:%m"
```

This line causes the **udev** device manager to run **multipath** every time a block device is added to the node. Even with this line removed, the **multipathd** daemon will still automatically create multipathed devices, and **multipath** will still be called during the boot process for nodes with multipathed root file systems. The only change is that multipathed devices will not be automatically created when the **multipathd** daemon is not running, which should not be a problem for the vast majority of multipath users.

5.7. ISSUES WITH QUEUE_IF_NO_PATH FEATURE

If a multipath device is configured with **features "1 queue_if_no_path"**, then any process that issues I/O will hang 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, use the **dmsetup** command to edit the policy at runtime for a particular LUN (that is, for which all the paths are unavailable). For example, if you want to change the policy on the multipath device **mpathc** from **"queue_if_no_path"** to **"fail_if_no_path"**, execute the following command.

```
dmsetup message mpathc 0 "fail_if_no_path"
```

Note that you must specify the **mpath*n*** alias rather than the path.

5.8. 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.

The dm status is similar to the path status, but from the kernel's point of view. The dm status has two states: **failed**, which is analogous to **faulty**, and **active** which covers all other path states. Occasionally, the path state and the dm state of a device will temporarily not agree.

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

NOTE

When a multipath device is being created or modified, the path group status, the dm device name, the write permissions, and the dm status are not known. Also, the features are not always correct.

5.9. MULTIPATH QUERIES WITH MULTIPATH COMMAND

You can use the **-l** and **-ll** options of the **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, there are three verbosity levels you can specify 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.

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
```

5.10. MULTIPATH COMMAND OPTIONS

Table 5.1, "Useful **multipath** Command Options" describes some options of the **multipath** command that you may find useful.

Table 5.1. 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.

5.11. 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.

The following command displays 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)
```

5.12. THE MULTIPATHD COMMANDS

The **multipathd** commands can be used to administer the **multipathd** daemon. For information on the available **multipathd** commands, see the **multipathd**(8) man page.

The following command shows 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. You can display a list of available wildcards with the following command.

```
# multipathd show wildcards
```

As of Red Hat Enterprise Linux release 7.3, the **multipathd** command supports new 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.

The following **multipathd** commands show 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 following **multipathd** commands show 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
```

The following commands show 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
```

5.13. 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, you can enter a interactive command, or you can enter **CTRL-D** to quit.

The **multipathd** interactive console can be used to troubleshoot problems you may be having with your system. For example, the following command sequence displays the multipath configuration, including the defaults, before exiting the console.

```
# multipathd -k
> > show config
> > CTRL-D
```

The following command sequence ensures that multipath has picked up any changes to the **multipath.conf**,

```
# multipathd -k
> > reconfigure
> > CTRL-D
```

Use the following command sequence to ensure that the path checker is working properly.

```
# multipathd -k
> > show paths
> > CTRL-D
```

5.14. CLEANING UP MULTIPATH FILES ON PACKAGE REMOVAL

If you should have occasion to remove the **device-mapper-multipath rpm**. file, note that this does not remove the **/etc/multipath.conf**, **/etc/multipath/bindings**, and **/etc/multipath/wwids** files. You may need to remove those files manually on subsequent installations of the **device-mapper-multipath** package.

APPENDIX A. REVISION HISTORY

Revision 7.1-1 **Wed Aug 7 2019** **Steven Levine**
Document version for 7.7 GA publication.

Revision 6.1-1 **Thu Oct 4 2018** **Steven Levine**
Document version for 7.6 GA publication.

Revision 5.1-1 **Thu Mar 15 2018** **Steven Levine**
Document version for 7.5 GA publication.

Revision 5.1-0 **Wed Dec 13 2017** **Steven Levine**
Document version for 7.5 Beta publication.

Revision 4.0-6 **Wed Jul 19 2017** **Steven Levine**
Document version for 7.4 GA publication.

Revision 4.0-3 **Wed May 10 2017** **Steven Levine**
Preparing document for 7.4 Beta publication

Revision 0.4-5 **Wed Nov 16 2016** **Steven Levine**
Small update to 7.3 version

Revision 0.4-3 **Mon Oct 24 2016** **Steven Levine**
Version for 7.3 GA publication

Revision 0.4-1 **Mon Aug 15 2016** **Steven Levine**
Preparing document for 7.3 Beta publication

Revision 0.3-6 **Mon Nov 9 2015** **Steven Levine**
Preparing document for 7.2 GA publication

Revision 0.3-3 **Wed Aug 19 2015** **Steven Levine**
Preparing document for 7.2 Beta publication.

Revision 0.2-7 **Mon Feb 16 2015** **Steven Levine**
Version for 7.1 GA release

Revision 0.2-6 **Thu Dec 11 2014** **Steven Levine**
Version for 7.1 Beta release

Revision 0.1-22 **Mon Jun 2 2014** **Steven Levine**
Version for 7.0 GA release

Revision 0.1-10 **Wed Apr 9 2014** **Steven Levine**
7.0 Beta update

Revision 0.1-3 **Tue Nov 26 2013** **Steven Levine**
Version for 7.0 pre-Beta

Revision 0.1-1 **Wed Jan 16 2013** **Steven Levine**
Branched from the Red Hat Enterprise Linux 6 version of the document

INDEX

Symbols

A

B

C

D

M

N